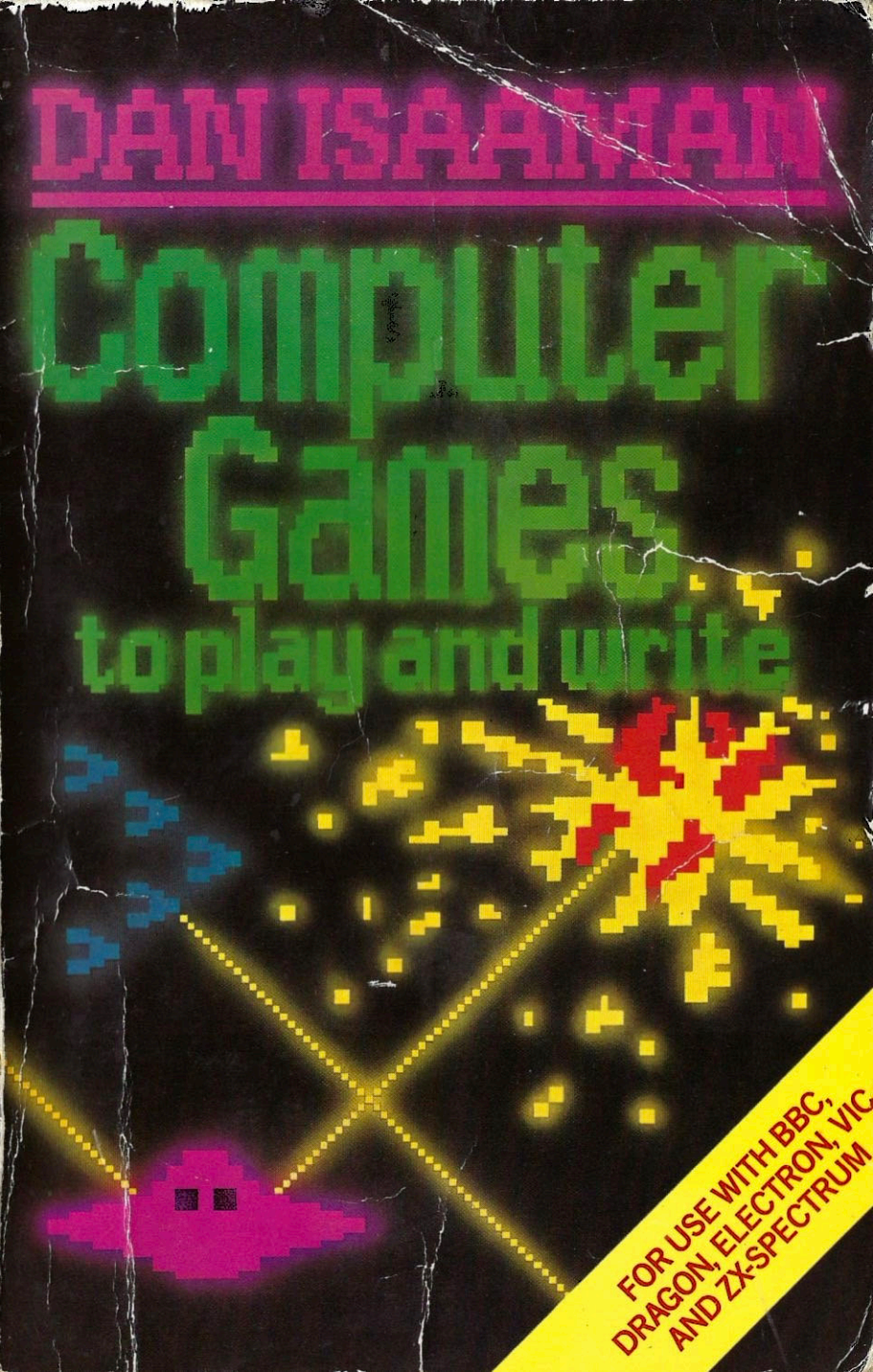
DAN ISAAMAN

Computer Games

to play and write



FOR USE WITH BBC, DRAGON, ELECTRON, VIC AND ZX-SPECTRUM

COMPUTER GAMES

to play and write

Dan Isaaman is currently studying Computer Engineering at Manchester University. He attended University College School in Hampstead and has been involved with computers ever since he was given a kit to build one at the age of fourteen.

While studying for 'A' levels in Maths, Further Maths, Physics and Chemistry, he wrote two books of computer games for Usborne. In his year off between school and university he wrote software for several large computer firms.

DAN ISAAMAN

Computer Games

to play and write

for use with ZX-Spectrum, BBC, Electron, VIC and Dragon computers

A Sparrow Book
Published by Arrow Books Limited
17-21 Conway Street, London W1P 6JD

An Imprint of the Hutchinson Publishing Group

London Melbourne Sydney Auckland
Johannesburg and agencies throughout the world

First published 1983

© Dan Isaaman 1983
Illustrations © Sparrow Books 1983

This book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent resold, hired out, or otherwise circulated without the publisher's prior consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser

Designed by Roger Walker/Paul Rogers

Set in Compugraphic Congress
by Angus Graham Associates Ltd

Made and printed in Great Britain
by the Guernsey Press Co Ltd
Guernsey, C.I.

ISBN 0 09 33330 9

I would like to thank Eleanor Nabney, without whom the programs in this book would not have been tested so thoroughly, and many would not even exist. I would also like to thank my trusty EPSON QX-10 computer, on which most of this book was written and printed, and my mother, who had to put up with the computers all over the house. Thanks are also due to Sinclair Research and Dragon Data for the loan of their computers.

To Eleanor
with love

CONTENTS

Foreword *9*

Notes for using the book *11*

Summary of instructions *15*

Game 1: Christmas Time *17*

Game 2: Red Alert! *21*

Game 3: Spy Codes *24*

Game 4: Ring-a-Ring O'Robot *30*

Game 5: Nim *33*

Game 6: From the Depths of Time *37*

Game 7: Galaxy Hitch-hike *41*

Game 8: Planet Lander *44*

Game 9: Animal *49*

Game 10: Silo Raid *53*

Game 11: Haunted House *58*

Game 12: Escape *62*

Game 13: Space Dogfight *66*

Game 14: Death Raid *70*

Game 15: Cowboys and Indians (ZX-Spectrum) *75*

Game 16: Cowboys and Indians (VIC) *80*

Game 17: Cowboys and Indians (BBC/Electron) *85*

Game 18: Cowboys and Indians (Dragon) *91*

How to write your own program: Sabotage *96*

BASIC and its commands *105*

Glossary *112*

Appendix *126*

I had great fun creating the games in this book. Some are games of skill, such as *Haunted House* and *Planet Lander*; others are mind-stretchers, such as *Christmas Time* and *Nim*; all, I hope, will provide you with hours of entertainment as you pit your wits against your computer. The games are designed to be used on five different computers, but at the end of the book you will find a supergame created especially for your machine.

Having played all the games in this book you may feel tempted to write some games yourself. Throughout the text I have put in tips and explanations on the computers and programs. There are also full explanations of what the program is doing line by line, and suggestions for ways in which you can change and adapt the games.

I do hope this book manages to fire your imagination and induce lots of midnight sessions at the computer, and I hope everyone ends up creating games that are at least as good as the ones in this book.

Happy computing!

Dan Isaaman

The purpose of this book is to provide you with exciting games to play on your computer. In the course of playing the games you will find that your skills and knowledge of programming increase. I hope that you may be inspired to write your own games.

The book is designed for use with five computers:

BBC
Dragon
Electron
VIC
ZX-Spectrum

There are slight differences in the way the BASIC language inside each of these computers works. Because of this, the programs have changes in some lines depending on which computer you are using. There are two areas of change:

1. At the end of each program you will see that there is a section written specially for your computer. When you get to this section, pick out the box that holds the lines for your computer and type only these lines into your computer.
2. Lines in the earlier part of the program sometimes need changing too. When this happens the lines are printed in green. A letter in the left-hand margin indicates which line you should type in. The letters are as follows:

```
b  BBC and Electron
s  ZX-Spectrum
d  Dragon
v  VIC
```

When you see these letters in the margin, type in only the line that has the letter corresponding to your computer against it. For instance, suppose you see the lines:

```
vbd 90 IF H$(Q)<>"" THEN GOTO 80
s   90 IF H$(Q,1)<>" " THEN GOTO 80
```

If you own a ZX-Spectrum, then you should type in line 90 as:

```
90 IF H$(Q,1)<>" " THEN GOTO 80
```

If you own a BBC, Dragon, Electron, or VIC you should type line 90 in as:

```
90 IF H$(Q)<>"" THEN GOTO 80
```

The programs in this book are written in BASIC. A section at the back of the book explains how this language works. Once you know this language you can use it to write simple programs and games. The game *Sabotage* shows you how you can create a game.

To type in and play a game, the first thing you must do is get your computer ready. Plug the computer and the television or monitor screen into the mains and connect them using the cable provided with your computer. Switch them on.

You are now ready to type in a program. The computer knows that you are typing in a program when each line that you type begins with a number. The computer sorts the individual lines into a program, with the lines appearing in numerical order. If you look at the programs in this book you will see that the line numbers usually go up in steps of ten. Programs are written in this way so that if there are any extra instructions to be added this can be done using the line numbers that are still available between those already used.

When you are typing in the program, be very careful to get it *exactly* right. Type in each line one by one, starting with the line number and finishing by pressing the RETURN or ENTER key. Make sure that the commas, semicolons, spaces and confusing letters and numbers such as O and 0 are typed in exactly as they are in the book.

If you have a tape recorder or disc drive connected to your computer, you can store the game program on tape or disc. If you do this, you will of course only need to type in a game program once, and you can load it back from tape or disc whenever you want. Look up the commands SAVE and LOAD in the manual for all the computers except Dragon. For Dragon look up CSAVE and CLOAD. This section will tell you how to give the program a name

on disc or tape so that you can load it back into the computer at any time.

Note that with some tape recorders it may take a little time to adjust the recorder and the computer to achieve accurate program storage. Your computer manual should provide full details.

Once you have typed in the program, read the scenario above the game that tells you what you must do once the game has started. Then type the command RUN into the computer. This is a direct command to the computer so it does not need a line number. Then press the RETURN or ENTER key and the computer will start the game.

The game should now work perfectly. If it does not, the computer will usually tell you immediately by stopping the program and displaying an error such as:

```
SYNTAX ERROR IN LINE 120
```

Alternatively, it may be that the computer just does not behave as you expect. In either case you will have caused a bug (error) in the program by typing in an instruction incorrectly. You must go through the program checking every single line against every single line of the program listed in this book. To get the program displayed on the screen you use the LIST command.

For example for the BBC and Electron,

```
LIST 10,200
```

will list lines 10 to 200.

For Dragon and VIC,

```
LIST 10-200
```

will list lines 10 to 200.

For ZX-Spectrum,

```
LIST 10
```

will list the program to the bottom of the screen.

The question Scroll? then appears. If you type in Y the listing will continue. If you type in N it will stop.

When you have found the mistake, type in your correction. The simplest way to change a line is to retype it, beginning with the line number. This automatically wipes out the previous version of the line. If you want to delete a line, type the line number immediately followed by the RETURN or ENTER key. You can use LIST at any time to check your corrections.

Once you have corrected your mistake, type RUN to start the game as before. Remember — if you have already saved the program on casette or disc you must now replace it with the corrected version.

At the end of each game I have suggested changes that you might like to put into the program. To make these changes to the games, type in the new lines just as you did the old. If you type in a line with a line number that is already in the program, the computer will simply replace the old line with your new one. If you wish to delete a line in the program, then, as before, just type in its line number immediately followed by the RETURN or ENTER key.

Read how to edit your program in the computer manual, so that you can make the changes quickly.

When typing in the games, you should always remember the following things:

1. Type each line exactly as it is printed in this book. Start with the line number and press the RETURN or ENTER key at the end of the line.
2. Make sure you type in the correct lines for your particular computer where there is a choice. When two or more lines are in green, with letters against them in the margin, you must type in only one. The indicator symbols and the corresponding computers are as follows:

```
b  BBC and Electron
s  ZX-Spectrum
d  Dragon
v  VIC
```

3. At the end of the program, type in the lines from the ruled box applicable to your computer.
4. Once you have typed in the program, you can store it if you have a tape recorder or disc drive. Then type RUN to start the game.
5. If the program does not work properly, LIST it and go through it, checking it against the listing in this book. Check every line for typing mistakes. Usually you will find that things like commas, semicolons, spaces and confusing letters and numbers such as `O` and `0` have been typed wrongly.

You are Father Christmas and you're in trouble. It is Christmas Eve and you have a large bag of wonderful presents to deliver to the children living in Muddlesome Street. But each parcel is only labelled with the family surname and not with their address.

There are ten houses in the street and ten presents in the bag. When you bring out a parcel you must choose which chimney to go down. Once you're down the chimney you're able to find out whose house it is and, if it is the wrong one, how far away the right house is. You must memorize this information. Dawn is approaching and if you can't sort the muddle out by then, you will surely turn into a reindeer.

```
    10 GOSUB 1000 : GOSUB 2000
    20 PRINT "CHRISTMAS TIME"
    30 PRINT
    40 LET N=10 : LET L=0
vbd 50 DIM H$(N)
s   50 DIM H$(N,10)
    60 FOR I=1 TO N
    70 READ P$
    80 LET Q=FNR(N)
vbd 90 IF H$(Q)<>"" THEN GOTO 80
s   90 IF H$(Q,1)<>" " THEN GOTO 80
   100 LET H$(Q)=P$
   110 NEXT I
   120 FOR G=1 TO N*2
   130 LET Q=FNR(N)
vbd140 IF H$(Q)="" THEN GOTO 130
s  140 IF H$(Q,1)=" " THEN GOTO 130
   150 PRINT "THE LABEL SAYS:";H$(Q)
   160 PRINT "HOUSE NUMBER ";
vbd170 INPUT Z
s  170 INPUT Z : PRINT Z
   180 IF Z<1 OR Z>N THEN GOTO 160
vbd190 IF H$(Z)="" THEN PRINT "YOU ALREADY
       DELIVERED ONE HERE" : GOTO 250
```

```
190 IF H$(Z,1)=" " THEN PRINT "YOU ALREADY
    DELIVERED ONE HERE" : GOTO 250
200 IF Z<>Q THEN GOTO 230
210 PRINT "PRESENT DELIVERED"
220 LET L=L+1 : LET H$(Q)="" : GOTO 250
230 PRINT "THIS HOUSE BELONGS TO ";H$(Z)
240 PRINT "THE PERSON YOU WANT IS ";ABS(Z-Q);
    " DOORS AWAY."
250 IF L=N THEN GOTO 290
260 NEXT G
270 PRINT "YOUR TIME RAN OUT. YOU TURNED INTO
    A REINDEER!"
280 GOTO 4000
290 PRINT "YOU DELIVERED THEM ALL IN ";G;
    " GOES. WELL DONE"
300 GOTO 4000
310 DATA "SMITH","JONES","BLOGGS","WILSON"
320 DATA "MILLER","ENGLAND","EDWARDS"
330 DATA "O'BRIEN","CLARKE","GROT"
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

Lines 10-30 clear the screen and print out the title.

Line 40 sets the number of houses to 10 (in N), the number of presents delivered to zero (in L).

Line 50 sets up an array for the names of the family in each house.

Lines 60-110 randomly put all the family names (in lines 310-330) into the array H$.

Line 120 is the start of the loop giving you 20 goes (with 10 houses in the road).

Lines 130-140 choose a random name for the parcel, and check that it hasn't already been delivered.

Line 150 prints a message telling you which name is on the parcel.

Lines 160-180 get your chosen house number and check that it is allowed.

Line 190 checks if there is already a present there.

Lines 200-220 check if you got the house number right, and if so, tell you; increase your score and delete the family name to show that the present has been delivered.

Lines 230-240 tell you whose house you have chosen and how far away the right one is.

Line 250 checks whether all the presents have been delivered, and if so, goes to 290.

Line 260 is the end of the loop giving you 20 goes.

Lines 270-280 print a message if you didn't deliver all the presents in time, and stop the program.

Line 290-300 print a message if you managed to deliver all the presents in time, and stop the program.

Lines 310-330 hold the names of the families in the street.

Lines 1000-4000 contain the standard routines.

Changes you can make:

1) To make the game a bit harder, you can get the computer to clear the screen after each go, so that you can't look back over your last few goes. To do this, add a new line at 125 to clear the screen:

```
125 GOSUB 2000
```

and add a delay (so that you can read the details about the house before they disappear) at line 255:

```
255 FOR I=1 TO 2000 : NEXT I
```

2) You can change the people's names by changing the strings in lines 310-330, and if you want to add more, add an extra DATA line after 330, with your extra names in quotes, sepatated by commas, and increase the number in line 40 from 10 to however many people you have.

The alarm in an underground atomic centre full of top security weapons has been set off accidentally. You have to discover the code to the lock on a door to get in to switch off the alarm system. If the alarm is not turned off quickly the whole complex with all its scientists and valuable equipment will be automatically sealed underground forever. You have only time for ten attempts at the two-digit code before the auto-sealing device is activated.

On each go you may move the combination lock a certain number of positions to the left or right. The dial has ten numbers on either side of the central zero. Your equipment can detect whether you have moved the dial past the correct number, because a louder click will be heard. Good luck!

```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "RED ALERT"
 30 PRINT
 40 LET G=0 : LET G1=10
 50 FOR I=1 TO 2
 60 LET P=FNR(21)-11
 70 LET P1=0
 80 PRINT "NO. TO TURN (LEFT IS -,RIGHT IS +)"
 90 INPUT T
100 IF ABS(T+P1)>10 OR T=0 THEN GOTO 90
110 FOR J=1 TO ABS(T)
120 PRINT "CLICK...";  : NEXT J : PRINT
130 LET P1=P1+T
140 IF P=P1 THEN GOTO 200
150 IF SGN(P1-P-T)=SGN(P1-P) THEN GOTO 170
160 PRINT "I THINK ONE CLICK WAS LOUDER"
170 LET G=G+1 : IF G<G1 THEN GOTO 80
180 PRINT "YOU HAVE BEEN BLOWN UP"
190 GOTO 4000
200 PRINT "CLUNK...YOU GOT IT!"
210 NEXT I
220 PRINT "WELL DONE.YOU CRACKED
    IT IN TIME!"
230 GOTO 4000
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

Lines 10-30 clear the screen and print the title of the game.

Line 40 sets G (your current number of tries) to zero, and G1 (the maximum number of tries) to ten.

Line 50 sets up the loop giving you two positions on the dial to solve.

Line 60 chooses a random position from -10 to 10 and saves it in P.

Line 70 sets your current position (in P1) to zero (ie the centre of the dial).

Lines 80-90 ask the player how much he wants to turn the dial, and gets the answer in T.

Line 100 if this is out of range, then asks again.

Lines 110-120 print out 'CLICK...' that many number of times.

Line 130 sets your new position in P1.
Line 140 checks whether you have got the right position and if so, goes to 200, which prints a message and repeats the loop.
Lines 150-160 check whether you passed the correct position while turning, and if so print a message.
Line 170 increases the count of the number of goes that you have had, and if you haven't finished, goes back for another try.
Lines 180-190 print a message if you ran out of tries, and stop the program.
Lines 200-210 print a message and loop back to give you the next position on the dial.
Lines 220-230 print a message if you managed to crack the combination, and stop then program.
Lines 1000-4000 contain the standard routines.

Changes you can make:

1) At the moment, the combination lock has a two-digit combination number, and for each digit, you always start at position zero (the centre of the dial). You can add more digits to the number by changing the number 2 in line 50 to a higher number, and if you need to, changing the number of goes allowed in line 40.

2) You can try leaving the dial in the same position when you get one digit, instead of moving it back to the centre (line 70 does this each time). If you delete line 70, and put instead a new line 45 LET P1=0 , then the centre position will only be set on the first digit.

3:SPY CODES

You are at the centre of an international spy ring. It is crucial that you get your messages out of a foreign country undetected. You decide to do this via a coding program. The coding program will turn your secret messages into unreadable rubbish. The agent at home will receive these messages on his computer which will decode them.

Type every word of your message into the computer separately. The computer will turn each word into code and flash it up onto the screen. You must then remember the coded word — agent's instructions forbid you to write down any coded message. When the screen clears, type the coded word back into the computer and it will be transmitted to the agent at home. Then move on to the next word in your message.

When the whole message is typed in in this way, type an * to indicate 'message complete'. You will then see whether you have been successful in getting the vital information through.

```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "SPY CODES"
 30 PRINT
 40 LET C=FNR(25)
 50 LET M$="" : LET N$=""
 60 PRINT "TYPE IN A WORD (UPPER CASE ONLY,
    * TO END):"
 70 INPUT W$
 80 IF W$="*" THEN GOTO 210
 90 LET M$=M$+W$+" "
100 GOSUB 500
110 GOSUB 2000
120 PRINT "CODED WORD IS:"
130 PRINT X$
```

```
  140 FOR I=1 TO LEN(W$)*200 : NEXT I
  150 GOSUB 2000
  160 PRINT "TYPE CODE TO TRANSMIT:"
  170 INPUT W$
  180 GOSUB 500
  190 LET N$=N$+X$+" "
  200 GOTO 60
  210 PRINT "YOU SENT THE MESSAGE:"
  220 PRINT N$
  230 PRINT "THE CORRECT ONE WAS:"
  240 PRINT M$
  250 IF M$=N$ THEN PRINT "WELL DONE!" : GOTO
      4000
  260 IF LEN(M$)<>LEN(N$) THEN GOTO 320
  270 LET N=0
  280 FOR I=1 TO LEN(M$)
vbd290 IF MID$(M$,I,1)<>MID$(N$,I,1) THEN LET
      N=N+1
s 290 IF M$(I)<>N$(I) THEN LET N=N+1
  300 NEXT I
  310 PRINT "YOU MADE ";N;" MISTAKES"
  320 PRINT "YOUR AGENT IS TOTALLY CONFUSED"
  330 GOTO 4000
  500 LET X$=""
  510 FOR I=1 TO LEN(W$)
vbd520 LET X=ASC(MID$(W$,I,1))+C
s 520 LET X=CODE(W$(I))+C
  530 IF X>90 THEN LET X=X-26
  540 IF X<65 THEN LET X=X+26
  550 LET X$=X$+CHR$(X)
  560 NEXT I
  570 LET C=-C
  580 RETURN
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

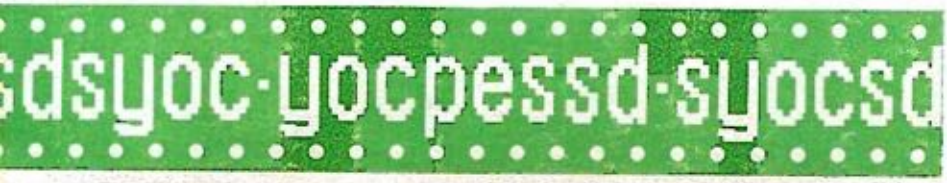
Lines 10-30 clear the screen and display the title of the game.

Line 40 chooses the secret code number for this game.

Line 50 makes the strings that will hold the message that you typed in (M$) and the final decoded message (N$) blank.

Lines 60-70 get a word from you, into W$.

Line 80 checks if you want to finish, and if so, the computer goes to line 210.



Line 90 adds this word onto the message, with a space after it.

Line 100 calls the subroutine that converts this message to code.

Lines 110-130 clear the screen and display the coded word on the screen.

Lines 140-150 wait for a time, depending on the length of the word, and then clear the screen again.

Lines 160-170 get the code word from you to transmit.

Lines 180-190 call the subroutine that will decode the word, and adds the decoded word onto the final message in N$, along with a space.

Line 200 repeats the process by going back to line 60, where you will enter a new word (or a '*').

Lines 210-240 display the final decoded message, and the original message that you wanted to send.

Line 250 checks whether you got it all right. If so, it displays a message and stops the program.

Line 260 checks if the length of the two messages is the same or not. If not, the computer will go to line 320 and print a suitable message.

Lines 270-300 go through the two strings, checking each letter. Each time a letter is wrong, the variable N is increased by one (starting at zero), so it will contain a count of the number of your mistakes.

Line 310 displays a message telling you how much you got wrong.

Lines 320-330 display a suitable message, and stop the program.

Lines 500-580 contain the subroutine to code and decode a word. The first time it is used, it will code the string in W$ and put the result in X$. The next time, it will decode the string in W$ and put the result in X$. It does this by changing the sign of the secret code number, C, in line 570. If effect, each letter is moved through the alphabet C times to get its coded version.

Lines 1000-4000 contain the standard routines.

Changes you can make:

1) The length of time that the coded word appears on the screen is controlled in line 140. If you want to make it faster, make the number 200 smaller. If you want to make it slower, increase the number.

2) You could make the computer produce the message on its own. You could put the message in a DATA statement at the end of the program, with each word in quotation marks, and the last word being a *. You could even have several messages in stored this way, and could select randomly from

them. Here is one way you could do this:

```
 42 LET M=FNR(3)
 44 FOR I=1 TO M
 46 READ W$ : IF W$<>"*" THEN GOTO 46
 48 NEXT I
 60 READ W$
delete line 70

600 DATA "*","THE","ANSWER","IS","FORTY",
    "TWO","*"
610 DATA "OPERATION","PINEAPPLE","TO","GO",
    "AHEAD","*"
620 DATA "WHAT","IS","FOR","LUNCH","*"
```

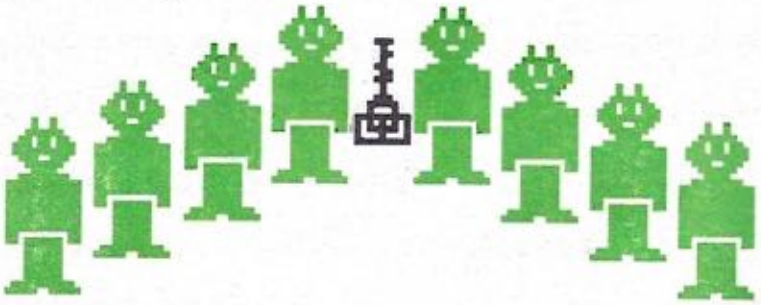
Lines 42-48 choose a random message from the three available, and read through the words until the start of that message (ie. if M is one, it will read the first * only, if M is two, it will read up to the second *, so that the program will start with the second message).

Line 60 gets the word of the message from the DATA statements.

Lines 600-620 contain the three messages. You can change these or add your own, but remember to change the number 3 in line 42 to the total number of messages.

Androids surround you. They have stolen the time key without which you cannot leave their planet. They are teasing you, passing the time key around the circle between themselves, constantly changing the direction it goes in. They've told you that if you can touch the android who holds the key, you can have it back. Each android is represented on the screen by a letter.

From the glimpses you get of the key as it goes round, try to touch the person holding it by typing in his letter as quickly as possible. If you aren't able to touch the android holding the key after a certain time, you'll just have to accept you'll never see the planet Earth again.



```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "RING-A-RING O'ROBOT"
 30 PRINT
b  40 LET C1=20 : LET C2=10
d  40 LET C1=16 : LET C2=8
vs 40 LET C1=11 : LET C2=10
b  50 LET D1=12 : LET D2=8
d  50 LET D1=8 : LET D2=6
vs 50 LET D1=10 : LET D2=8
 60 FOR N=0 TO 11
 70 LET P$=CHR$(65+N) : GOSUB 210
 80 NEXT N
 90 LET N=FNR(12)-1
100 LET D=1
110 IF FNR(10)>4 THEN GOTO 170
120 LET P$="*" : GOSUB 210
130 LET J$="" : FOR I=1 TO 100 : GOSUB 3000
```

```
   140 LET J$=J$+I$ : NEXT I
vbd150 IF ASC(J$+" ")-65=N THEN PRINT "GOT IT!"
       : GOTO 4000
s  150 IF CODE(J$+" ")-65=N THEN PRINT "GOT IT!"
       : GOTO 4000
   160 LET P$=CHR$(65+N) : GOSUB 210
   170 LET N=N+D : IF FNR(10)=1 THEN LET D=-D
   180 IF N<0 THEN LET N=N+12
   190 IF N>11 THEN LET N=N-12
   200 GOTO 110
   210 LET X=C1+INT(SIN(N/6*3.142)*D1)
   220 LET Y=C2-INT(COS(N/6*3.142)*D2)
b  230 PRINT CHR$(31)CHR$(X)CHR$(Y);P$;
s  230 PRINT AT X,Y;P$;
d  230 PRINT @(X+32*Y),P$;
v  230 PRINT CHR$(19); : FOR Q=1 TO Y : PRINT :
       NEXT Q : PRINT TAB(X);P$;
   240 RETURN
```

For the different computers

BBC / Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

Lines 10-30 clear the screen and print the title.

Lines 40-50 set up the variables that control the size of the screen. C1 and C2 hold the coordinates of the centre of the ring, while D1 and D2 hold the size.

Lines 60-80 display the ring of 12 letters on the screen.

Line 90 chooses the starting point for the time key.

Line 100 sets the direction of the movement in D to clockwise (anticlockwise is -1).

Line 110 decides whether the time key should be shown on the screen. If not, the computer goes to line 170.

Line 120 displays a * at the current position in the ring.

Lines 130-140 check if the player is pressing a key.

Line 150 checks if you got the right letter. If so, displays the message 'GOT IT' and stops.

Line 160 re-displays the letter in the ring.

Line 170 moves the position of the time key. Sometimes, the direction of movement is changed.

Lines 180-190 checks if the position has gone out of range. If so, it is re-calculated.

Line 200 goes back to repeat the loop.

Lines 210-220 work out the position on the screen, given the number that the key is at.

Lines 230-240 display the character at that position, and return to the main program.

Lines 1000-4000 contain the standard routines.

This game is of Chinese origin and is one of the world's oldest number games. It's a game for two people and you can play it against your computer.

It has very simple rules. The computer sets up several piles of matches. Each pile has a random number of matches in it. You and the computer take alternate turns. On each turn the player can take any number of matches from any one pile. The winner is the player who is able to remove the last pile from the table in his turn.

As this is a game of numbers and logic, and computers are best at these things, you will find the computer is VERY good at playing. But there are ways to win ... Good luck!

```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "NIM"
 30 PRINT
 40 LET N=FNR(4)+2
 50 DIM P(N)
 60 FOR I=1 TO N : LET P(I)=FNR(7)
 70 NEXT I : GOSUB 400
 80 PRINT "DO YOU WANT TO START (Y OR N)?"
 90 GOSUB 3000 : IF I$="" THEN GOTO 90
100 IF I$="Y" OR I$="y" THEN GOTO 210
110 LET C=0
120 FOR A=1 TO N : IF P(A)=0 THEN GOTO 160
130 FOR B=1 TO P(A) : GOSUB 500
140 IF C=1 THEN GOTO 190
150 NEXT B
160 NEXT A
```

```
170 LET A=FNR(N) : IF P(A)=0 THEN GOTO 170
180 LET B=FNR(P(A))
190 LET P(A)=P(A)-B : GOSUB 400
200 IF T=0 THEN PRINT "I WIN...." : GOTO 4000
210 PRINT "WHICH PILE"
220 INPUT A
230 IF A<1 OR A>N THEN GOTO 220
240 IF P(A)=0 THEN GOTO 220
250 PRINT "HOW MANY"
260 INPUT B
270 IF B<1 OR B>P(A) THEN GOTO 260
280 LET P(A)=P(A)-B : GOSUB 400
290 IF T=0 THEN PRINT "YOU WIN..." :
    GOTO 4000
300 GOTO 110
400 GOSUB 2000 : PRINT "PILE NO.","MATCHES"
410 LET T=0 : FOR I=1 TO N : PRINT I,P(I)
420 LET T=T+P(I) : NEXT I
430 RETURN
500 LET P(A)=P(A)-B
510 LET Q=0 : LET R=0 : LET S=0
520 FOR I=1 TO N : IF P(I)=0 THEN GOTO 550
530 RESTORE : FOR J=1 TO P(I) : READ X,Y,Z :
    NEXT J
540 LET Q=Q+X : LET R=R+Y : LET S=S+Z
550 NEXT I
560 LET P(A)=P(A)+B
570 IF Q/2=INT(Q/2) AND R/2=INT(R/2) AND S/2=
    INT(S/2) THEN LET C=1
580 RETURN
600 DATA 0,0,1,0,1,0,0,1,1,1,0,0,1,0,1,1,1,0,
    1,1,1
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

Lines 10-30 clear the screen and print the title.

Lines 40-50 choose the number of piles, and make an array variable to hold them.

Lines 60-70 fill each pile with a random number of matches and display the piles on the screen

Lines 80-100 ask whether you wish to start, and if you do, the program goes to 210.

Lines 110-160 control the computer's turn. If it finds a good move, it goes to line 190, otherwise ...

Lines 170-180 pick a random move for the computer.

Line 190 takes the matches away from the right pile, and displays the piles.

Line 200 checks whether the computer has won (if all the matches have gone).

Lines 210-270 get the player's move and make sure it is valid.
Line 280 takes the matches away from the right pile, and displays the piles.
Line 290 checks whether the player has won.
Line 300 loops back for the computer's next go.
Lines 400-430 display the piles on the screen.
Lines 500-600 contain the program to work out the computer's move.
Lines 1000-4000 contain the standard routines.

Changes you can make:

1) You can change the number of piles allowed in line 40. At the moment, there can be any number between 3 and 6. The maximum number of matches in each pile can be changed by changing the number 7 in line 60.

2) You could change the routine that displays the piles of matches (lines 400-430) so that it prints a row of *'s for each pile, where each star represents a match. One way to do this would be:

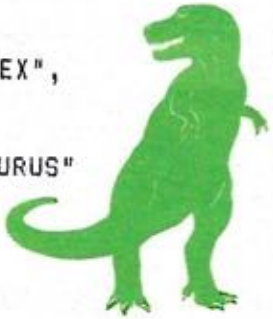
```
410 LET T=0 : FOR I=1 TO N : PRINT I,
415 FOR J=1 TO P(I) : PRINT "*"; : NEXT J
420 PRINT : LET T=T+P(I) : NEXT I
430 RETURN
```

If you are interested in maths, see if you can work out how the computer plays, from the program at lines 500 onwards. Hint - the DATA statement contains 3-digit binary numbers from 1 to 7. Otherwise, watch how it plays and see if you can learn from it - the computer is a very good player!

You have been sent back in time on a mission of exploration. You have a crew of four scientists aboard. Suddenly you find that your time machine is being attacked by dreadful prehistoric monsters. Your only chance of survival is to send the crew out one by one against the monsters. Each member of the crew is equipped with various methods of defence. But the monsters attack so quickly that you only have time to order the man to use one method of defence, before he is completely overwhelmed by the ferocious beast. Decide how the man is to defend himself by pressing the appropriate number key each time a monster attacks ... and discover whether you will survive.

```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "THE DEPTHS OF TIME"
 30 LET K=0 : LET K1=10 : LET P=FNR(3)
 40 FOR M=1 TO 4
 50 RESTORE : READ N : LET W=FNR(N)
 60 FOR I=1 TO W
 70 READ M$ : NEXT I : LET W=W+P
 80 IF W>3 THEN LET W=W-3 : GOTO 80
 90 FOR I=1 TO 1000 : NEXT I : GOSUB 2000
100 PRINT "YOU ARE BEING ATTACKED BY A"
110 PRINT M$
120 PRINT "1) FIRE POISON DART"
130 PRINT "2) FIRE HARPOON"
140 PRINT "3) ATTACK WITH KNIFE"
150 PRINT "4) RUN BACK TO SHIP"
160 PRINT "CHOOSE OPTION....."
170 FOR I=1 TO 150 : GOSUB 3000
180 IF I$>"0" AND I$<"5" THEN
    LET I=150
190 NEXT I : LET D=VAL("0"+I$)
200 IF D=4 AND
    FNR(10)<5 THEN
    PRINT "PHEW!":
    GOTO 50
210 IF D<>W THEN
    GOTO 260
```

```
220 PRINT "YOU KILLED IT" : LET K=K+1
230 IF K<K1 THEN GOTO 50
240 PRINT "WELL DONE,YOU DID IT!"
250 GOTO 290
260 PRINT "THE MONSTER KILLED ONE OF YOUR  CREW"
270 NEXT M
280 PRINT "OH DEAR,YOU HAVE ALL BEEN KILLED"
290 PRINT "YOU MANAGED TO KILL ";K;" MONSTERS"
300 GOTO 4000
500 DATA 5
510 DATA "TYRANNOSAURUS REX",
    "BRONTOSAURUS"
520 DATA "DIPLODOCUS",
    "STEGOSAURUS","ALIOSAURUS"
```



For the different computers

BBC / Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

Lines 10-20 clear the screen and print the title of the game.

Line 30 sets the number of monsters killed so far to zero (in K) and the total number of monsters that are going to attack you to 10 (in K1). It also sets P to a random number between 1 and 3. This is used to decide which weapon will kill which monster.

Line 40 is the start of the loop giving you four lives (one for each of your men).

Line 50 restores the pointer to the beginning of the data in lines 500 to 520, reads the number of different monster names into N, and picks a random monster in W.

Lines 60-70 read that name into M$ from the data in lines 510-520, and also add P to the chosen monster number, to decide which weapon will kill the monster.

Line 80 If W is greater than three, the computer takes away three until it is small enough.

Lines 90-160 delay a while, clear the screen and print out a message telling you that you are being attacked by the monster and asking you what to do.

Lines 170-190 get a decision from the player, if he is quick enough. He is only allowed to type the keys 1 to 4. Line 190 works out which key was pressed, and puts the number in D.

Line 200 lets you run away, sometimes!

Line 210 checks if you got the right weapon or not. If not, you lose a life.

Lines 220-230 tell you that you killed the monster, increases the count of monsters killed, and if there are any more left, goes back for another.

Lines 240-250 print a message when you kill all the monsters, and then goes to line 290, which

prints how many monsters you killed, and stops.

Lines 260-270 tell you that one of your men was killed, and then loops back for the next man, until all four men are dead...

Lines 280-300 tell you when all your men are dead, and how many monsters you managed to kill. The program then stops.

Line 500 holds the number of monster names in the following data statements.

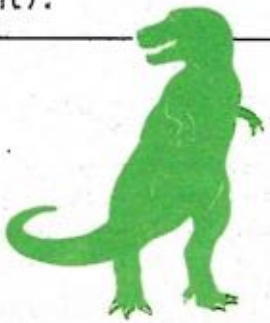
Lines 510-520 hold the different monster names.

Lines 1000-4000 contain the standard routines.

Changes you can make:

1) You can make the program easier or harder by changing the number of monsters that attack you (the number 10 in line 30) and the number of men in your crew (the number 4 in line 40).

2) You can easily add more monster types, by adding extra lines after line 520, beginning with DATA and with the names in quotes. Remember to change the number in line 500 to the total number of monster names (it is 5 at the moment).



You have been abandoned in the far reaches of the galaxy, and you have vital information about an underground organization planning to overthrow Galactic High Command. You must return to Earth in time to warn them of this plot. Your only hope of getting back is to hitch lifts from passing trade spaceships.

There are many passing ships, but can you manage to plan your way back to Earth in time to save the Empire. You have a limited amount of money with which you can try to bribe the captain of a ship to take you nearer to Earth, but you may just have to rely on luck and intuition...

```
  10 GOSUB 1000 : GOSUB 2000
  20 PRINT "GALAXY HITCH-HIKE"
  30 PRINT
vbd40 DIM S$(10)
s  40 DIM S$(10,8)
  50 LET M=2 : LET T=0
  60 FOR I=1 TO 10 : READ S$(I)
  70 NEXT I
  80 LET S=8+FNR(2)
  90 FOR I=1 TO 1000 : NEXT I :
     GOSUB 2000 : GOSUB 500
 100 PRINT "YOU ARE ON ";S$(S)
 110 LET T=T+1
 120 LET P=FNR(S)+FNR(10-S)*(FNR(2)-1)
 130 IF P=S THEN GOTO 110
 140 PRINT "THERE IS A SHIP GOING TO"
 150 PRINT S$(P)
 160 PRINT "DO YOU WANT TO TAKE IT?"
 170 FOR I=1 TO 500 : GOSUB 3000 : IF I$<>""
     THEN LET I=500
 180 NEXT I : IF I$="" THEN PRINT "IT LEFT!" :
     GOTO 90
 190 IF I$="Y" OR I$="y" THEN LET S=P
 200 IF S=1 THEN GOTO 300
 210 IF M=0 THEN GOTO 90
```

```
220 PRINT "DO YOU WANT TO TRY TO BRIBE
    THE CAPTAIN?"
230 FOR I=1 TO 500 : GOSUB 3000 : IF I$<>""
    THEN LET I=500
240 NEXT I : IF I$="" THEN PRINT "TOO LATE!" :
    GOTO 90
250 IF I$<>"Y" AND I$<>"y" THEN GOTO 90
260 IF FNR(10)<7 THEN PRINT "DIDN'T HELP!" :
    GOTO 90
270 LET S=S-1 : LET M=M-1
280 PRINT "YOU ARE NOW GOING TO"
290 PRINT S$(S) : IF S<>1 THEN GOTO 90
300 PRINT "YOU MADE IT TO EARTH!"
310 IF T>12 THEN PRINT "BUT THE
    REVOLUTION HAS STARTED"
320 IF T<=12 THEN PRINT
    "YOU GOT THERE IN TIME!"
330 PRINT "YOUR JOURNEY TOOK ";T;" WEEKS"
340 GOTO 4000
500 PRINT " *";TAB(20);"*"
510 PRINT "CENTRUS   *";TAB(16);"DELTAR"
520 PRINT TAB(8);"VEGA"
530 PRINT "*";TAB(15);"*"
540 PRINT "TRALIN";TAB(13);"SIRTEP"
550 PRINT TAB(10);"+"
560 PRINT TAB(9);"EARTH   *"
570 PRINT TAB(4);"*";TAB(16);"FROP"
580 PRINT " GRAPLOR";TAB(13);"*"
590 PRINT TAB(12);"DORF   *"
600 PRINT TAB(17);"PLOT"
610 RETURN
700 DATA "EARTH","SIRTEP","FROP","GRAPLOR"
710 DATA "VEGA","TRALIN","DORF","DELTAR"
720 DATA "CENTRUS","PLOT"
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

8: PLANET LANDER

You are on a secret mission for High Command in a prototype spaceship. After a five-month journey, you have come into orbit around the planet that is your destination. You must now try to land on it. Your instruments give you readings about the planet, telling you how strong the gravity is and what the atmosphere is like. You must land safely on the surface.

At each stage of the descent you type in how much you want your engines to thrust, in order to slow you down. As you get closer to the planet beware of going too fast. To land safely you need to go very slowly. But if you thrust too much you will start going upwards, towards sure execution by High Command who will not tolerate the failure of missions.

You only have a limited amount of fuel. You are able to burn from 0 to 40 units of fuel on each go.

```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "PLANET LANDER"
 30 PRINT
 40 READ N : LET P=FNR(N)
 50 FOR I=1 TO P : READ P$,G,A : NEXT I
 60 PRINT "YOU ARE COMING DOWN ONTO ";P$
 70 PRINT "GRAVITY IS ";G
 80 PRINT "AND AIR RESISTANCE IS ";
 90 IF A<4 THEN PRINT "LOW"
100 IF A>3 AND A<7 THEN PRINT "MEDIUM"
110 IF A>6 THEN PRINT "HIGH"
120 LET T=0 : LET H=250
130 LET V=30 : LET F=100
140 FOR I=1 TO 1500 : NEXT I
140 FOR I=1 TO 2500 : NEXT I
150 GOSUB 2000
160 PRINT "TIME ";T,"HEIGHT ";H
170 PRINT "SPEED ";V,"FUEL ";F
180 IF F=0 THEN GOTO 220
190 PRINT : PRINT "FUEL TO USE"
200 INPUT F1
```

```
210 IF F1<0 OR F1>40 THEN PRINT "CHEAT!" :
    GOTO 200
220 IF F1>F THEN LET F1=F
230 LET V1=V-F1+G : LET F=F-F1
240 LET D=(V1+V)/2
250 IF D>=H THEN GOTO 310
260 LET H=H-D
270 LET V=V1 : LET T=T+1
280 LET Q=INT(V*V/1800)*A
290 IF Q>10 THEN GOTO 360
300 GOTO 150
310 LET V1=V+(G-F1)*H/V
320 IF V1>7 THEN PRINT "YOU MADE
    A BIG CRATER! BAD LUCK"
330 IF V1>2 AND V1<=7 THEN PRINT
    "YOU BROKE YOUR LEG!"
340 IF V1<=2 THEN PRINT
    "GOOD ENOUGH LANDING!"
350 GOTO 4000
360 PRINT "YOU ARE GOING SO FAST,
    YOU WILL SOON BURN UP..."
370 LET A=A+10
380 IF Q<20 THEN GOTO 140
390 PRINT "SIZZLE.....SIZZLE
    ......BOOOOMMM!"
400 PRINT "YOU HAVE TURNED
    INTO A FALLING  STAR!"
410 GOTO 4000
500 DATA 8
510 DATA "VEGA",4,5
520 DATA "SNURG",8,8
530 DATA "GLIRP",2,1
540 DATA "DORTON",9,4
550 DATA "SPROG",6,6
560 DATA "FLORP",1,3
570 DATA "PLURG",7,9
580 DATA "RITAL",9,2
```

For the different computers

BBC / Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

lines 10-30 Clears the screen and prints the title.

line 40 Reads the number of planets available and selects a random number between 1 and that number.

line 50 Gets details of the selected planet by reading through the data P times. At the end of the loop, P$ will contain the selected planet's name, G its gravity and A the air resistance.

lines 60-80 These lines print out the description of the planet for the player.

lines 90-110 Print LOW, MEDIUM or HIGH for the air resistance depending on its value.

lines 120-130 Set up the variables for the game. T is time,H is height above the surface, V is speed and F is the amount of fuel you have.

lines 140-170 Delay a while, clear the screen and print out the values of T,H,V and F. This is where the computer starts each go.

line 180 If you have run out of fuel, the program skips past the INPUT of fuel.

lines 190-200 Get amount of fuel to use from the player.

line 210 Checks that he's not using a minus amount, or too much.

line 220 Checks that he's not using more than he has, and if so, use only what there is left.

lines 230-240 Calculate the new speed, new fuel, and distance moved (in D).

line 250 This checks whether the player has landed. If so, GOTO 310.

lines 260-270 Work out new height, put new speed into V and increase time by one.

line 280 Works out air resistance factor.

line 290 Is it critical? If so, GOTO 360.

line 300 Go back for next turn.

line 310 Works out speed when landed.

lines 320-340 Print a message depending on your final speed.

line 350 Stops the game.

line 360 Prints a warning message because you are heating up from air resistance.

line 370 Increases air resistance so that next time, Q will be much larger.

line 380 Go back for next go only if Q<20.

lines 390-400 You have burnt up!

line 410 Stops the game.

lines 500 DATA statements with the names and details of each planet. The first data statement has the number of planets in the program.

Lines 1000-4000 contain the standard routines.

Changes you can make:

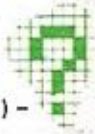
1) Lines 10-130, and line 500 onwards, set up all the different factors about the game. Most of these you can change yourself if you wish. In line 120, change the value of H if you want to start at a different height. Change V in 130 for a different starting speed, and F for a different amount of fuel (if you find it difficult to win, give yourself lots of fuel for practise until you get better, but remember, you're CHEATING!!!)

2) You can change or add planets in the game. Each line after line 500 has the planet name in quotes, followed by the gravity number, and then the air resistance number. If you want to add more planets, put them starting at line 590, and increase the number in line 500 (it starts at 8 since there are 8 planets in the list).

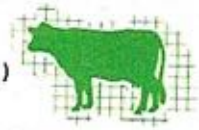
In this game the computer will try to guess what animal you are thinking of. The fun of the game is teaching the computer to ask more and more questions about different animals.

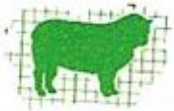
On each go, you are asked to think of an animal but not to tell the computer what it is. Then it will ask questions about the animal to which you must answer 'Yes' or 'No'. After a few questions, it will take a guess at the animal. If it is right, you will be glad you have such a clever machine. If it is wrong, you must teach it how to recognize the animal you were thinking of, so that next time it may be able to guess it correctly.

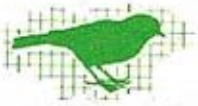
```
   10 GOSUB 1000 : GOSUB 2000
   20 PRINT "ANIMAL"
   30 PRINT
d  35 CLEAR 5000
vbd40 DIM Q$(50),N(50),A$(50)
s  40 DIM Q$(50,25) : DIM N(50) : DIM A$(50,15)
   50 READ N1,N2
   60 FOR I=1 TO N1
   70 READ Q$(I),N(I)
   80 NEXT I
   90 FOR I=1 TO N2
   100 READ A$(I) : NEXT I
   110 GOSUB 2000 : PRINT "THINK OF AN ANIMAL"
   120 LET P=1
   130 PRINT Q$(P)
   140 GOSUB 800
   150 IF I$="Y" OR I$="y" THEN LET P=N(P)-
       INT(N(P)/100)*100
   160 IF I$="N" OR I$="n" THEN LET P=INT(N(P)/
       100)
vbd170 IF LEFT$(Q$(P),1)<>"," THEN GOTO 130
s  170 IF Q$(P,1)<>"," THEN GOTO 130
vbd180 LET A=VAL(MID$(Q$(P),2))
s  180 LET A=VAL(Q$(P,2 TO))
   190 PRINT "IS IT A ";A$(A)
   200 GOSUB 800
   210 IF I$="N" OR I$="n" THEN GOTO 240
```

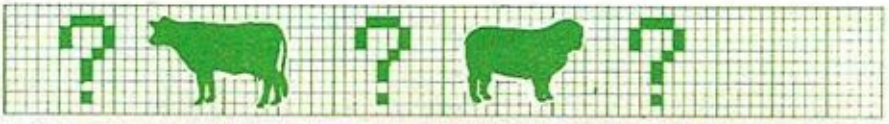


```
220 PRINT "WHY NOT TRY A DIFFERENT ANIMAL?"
230 GOTO 120
240 PRINT "THE ANIMAL YOU WERE THINKING OF
    WAS A"
250 INPUT T$
260 PRINT "TYPE IN A QUESTION THAT WOULD"
270 PRINT "TELL A ";T$;" FROM A ";A$(A)
280 INPUT S$
290 PRINT "FOR A ";T$;" THE ANSWER WOULD BE"
300 GOSUB 800 : LET Z=0 : IF I$="N" OR I$="n"
    THEN LET Z=1
310 LET Q$(N1+1)=Q$(P)
320 LET Q$(N1+2)="."+STR$(N2+1)
330 LET Q$(P)=S$
340 LET N(P)=N1*101+Z*99+102
350 LET N1=N1+2
360 LET A$(N2+1)=T$
370 LET N2=N2+1
380 GOTO 120
800 PRINT "? ";
810 GOSUB 3000
820 IF I$<>"Y" AND I$<>"N" AND I$<>"y" AND
    I$<>"n" THEN GOTO 810
830 PRINT I$
840 RETURN
900 DATA 3,2
910 DATA "DOES IT SWIM",302
920 DATA ".1",0,".2",0
930 DATA "PIKE","CROW"
```









For the different computers

BBC / Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

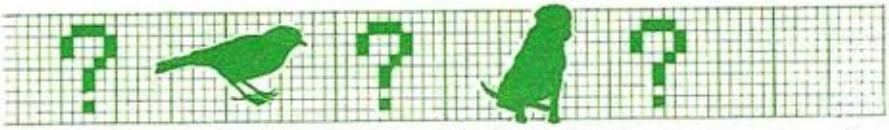
Lines 10-30 clear the screen and display the title of the game

Line 40 sets up three array variables. Q$ holds the questions, N holds the references to the next question each time, and a$ holds the different animal types.

Line 50 reads in the number of items in the question array, and the number of animals (into N1 and N2 respectively)

Lines 60-100 read in all the data in line 900 onwards into their respective arrays.

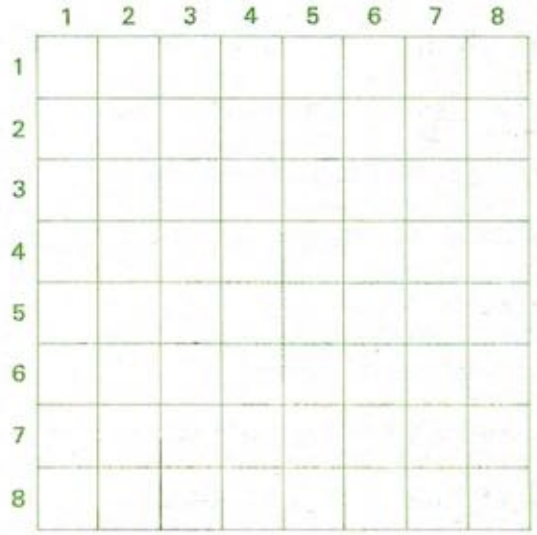
Line 110 clears the screen and prints the message: 'THINK OF AN ANIMAL'.



Line 120 sets the current question to number 1.
Line 130 prints the question
Line 140 gets an answer from you - either 'Y' or 'N'.
Lines 150-160 calculate the position of the next question or solution, from your answer and from the value stored in the array N.
Line 170 checks to see if there is a solution (ie the computer has run out of questions. If not, the computer will go back to print the next question.
Lines 180-190 gets the animal number and print 'IS IT A ....'
Line 200 gets your reply, either 'Y' or 'N'
Line 210-230 checks if if you answered 'N', and if not , prints a message and goes back for another game.
Lines 240-250 ask you to input the animal you were thinking of, and store it in T$.
Lines 260-280 ask for a question that would distinguish the two animals, and stores it in S$
Lines 290-300 ask you whether the answer to your question would be yes or no. If it is yes, then Z is set to 0, if no, Z is set to 1.
Lines 310-370 update the arrays and counters, so that the new animal is learned by the computer.
Lines 380-390 tells you that the computer has learnt the animal, and is ready to play again, which it does by going back to line 120.
Lines 800-830 accept a single key input from you, which can either be 'Y' or 'N', and print it on the screen.
Lines 900-930 contain the starting data for the program.
Lines 1000-4000 contain the standard routines.

You are sailing over enemy waters and you have been ordered to bomb and destroy all the underwater silos containing the enemy's newest and most dangerous weapons. Your ship contains twelve guided depth charges which will land on the sea bed at the spot you choose by giving co-ordinates.

You also have the use of a sonar but, unfortunately, it is not functioning correctly. The picture it gives is muddled and slightly incorrect. Can you make some sense from the display it gives to help you find the exact location of the silos you must destroy?




```
    10 GOSUB 1000 : GOSUB 2000
    20 DIM B(8,8)
    30 LET N=6
    40 LET H=0
v   50 LET T=10
bds 50 LET T=16
    60 FOR I=1 TO N
    70 LET X=FNR(8) : LET Y=FNR(8)
    80 IF B(X,Y)<>0 THEN GOTO 70
```

```
 90 LET B(X,Y)=-1 : NEXT I
100 LET Q$="SILO RAID"
110 FOR G=1 TO N*2
120 GOSUB 2000
130 PRINT Q$
140 PRINT "YOUR MAP";TAB(T);"SONAR"
150 LET F=FNR(7)
160 FOR Y=1 TO 8
170 LET Z=0 : GOSUB 400 : PRINT TAB(T);
180 LET Z=1 : IF Y+F>8 THEN LET F=F-8
190 GOSUB 400 : PRINT
200 NEXT Y
210 PRINT "X-POSITION" : INPUT X
220 PRINT "Y-POSITION" : INPUT Y
230 IF X<1 OR Y<1 OR X>8 OR Y>8 THEN GOTO 210
240 IF B(X,Y)=9 THEN GOTO 210
250 IF B(X,Y)=0 THEN LET B(X,Y)=9 : LET
    Q$="MISS"
260 IF B(X,Y)=-1 THEN LET B(X,Y)=1 : LET
    H=H+1 : LET Q$="HIT"
270 IF H=N THEN PRINT "ALL SILOS DESTROYED!" :
    GOTO 4000
280 NEXT G
290 PRINT Q$
300 PRINT "YOU RAN OUT OF DEPTH CHARGES"
310 PRINT "YOU HIT ";H;" SILOS"
320 GOTO 4000
400 FOR X=1 TO 8
410 LET V=B(X,Y+F*Z)
420 IF V=0 OR (V=-1 AND Z=0) THEN LET P$="."
430 IF V=1 OR (V=-1 AND Z=1) THEN LET P$="S"
440 IF V=9 THEN LET P$="*"
450 IF Z=1 AND FNR(10)>4 THEN LET P$="."
460 PRINT P$;
470 NEXT X
480 RETURN
```

For the different computers

BBC / Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

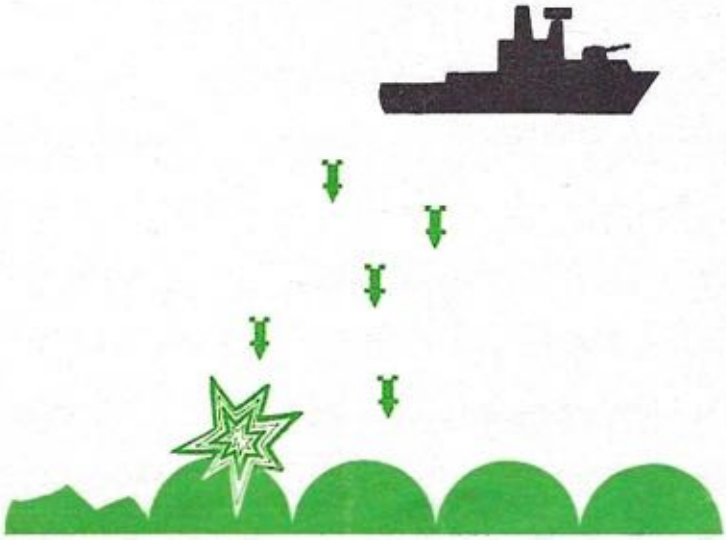
```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```



How the program works:

Line 10 clears the screen and sets up the random number generator.

Lines 20-50 set up the various variables in the program. The map is held in an array called B, which is a grid, eight by eight in size. N holds the total number of silos, H the number you have hit, and T and S$ are used for printing out the map on your screen.

Lines 60-90 put the silos in random positions on the map.

Line 100 puts the title of the game into Q$, which is printed in line 130. Q$ is then used to print the HIT and MISS messages.

Line 110 is the start of the loop giving you a number of goes equal to twice the number of silos you have to bomb.

Lines 120-140 clear the screen, prints the message in Q$ (see line 100 and lines 250-260) and prints the heading for the two maps.

Lines 150-200 control the printing of the two maps on the screen, along with the subroutine at line 300. F is used to muddle up the sonar map.

Lines 210-230 get the coordinates of your bombin position and check whether it is valid.

Line 240 checks whether there already is a bomb there, in which case you have to try somewhere else.

Lines 250-260 check if it is a miss or a hit, and change the numbers stored in the map array accordingly. If it is a hit, the count in H is increased. Q$ is set to a message that will be printed out next go, in line 130.

Line 270 checks if all the ships are hit, and if so tells you and stops the game.

Line 280 is the end of the loop, which goes back to print the new maps and gives you another go, until you have used all your bombs:

Lines 290-320 print a message when you have used up all your bombs, tell you how many silos you hit, and stop the game.

Lines 400-480 are a subroutine that prints out one section of one of the maps on the screen. Eight characters are printed out, depending on that row of the map stored in B, and depending on the variables Z and F which are used to muddle the sonar picture.

Lines 1000-4000 contain the standard routines.

Changes you can make:

1) You could change the number of silos by changing the number 6 in line 30.

2) Once you get better at the game, you can make the sonar picture less useful by changing the number 4 in line 450 to a smaller number. If the game is too difficult, however, increase this number to 5 or 6.

See if you can work out how the computer muddles the sonar picture, and try working out a different way to muddle it.

Can you banish the notorious ghosts from Shodthorn Manor? Its owners, who have been driven out by the ghosts, have employed you to get rid of all the ghosts in the Manor's ten spacious rooms. Fortunately, you met an old woman in the nearby village, who recited a spell to you that causes one ghost to vanish for ever every time the spell is said. But it had to be said only once for each ghost. If you say the spell too many times in a room a new ghost will be created, destined to haunt the house for ever. If you don't say it enough times, some of the old ghosts will remain.

Each time you enter a room, there will be a number of ghosts displayed all over the screen. You only have seconds to press a number key to tell the computer how many times you want to say the magic spell. You must try to say the spell the exact number of times as the number of ghosts in the room. Can you banish all the ghosts, or will there be some left haunting the house, for the next person to cope with? That next person could be you...

```
   10 GOSUB 1000 : GOSUB 2000
   20 PRINT "HAUNTED HOUSE"
   30 PRINT
   40 LET T=0
   50 FOR R=1 TO 10
   60 GOSUB 2000
   70 PRINT "ROOM NUMBER ";R
   80 LET N=FNR(8)
   90 FOR G=1 TO N
  100 LET X=FNR(19) : LET Y=FNR(15)
b 110 PRINT TAB(X,Y);"O O";
s 110 PRINT AT Y,X;"O O";
d 110 PRINT @(Y*32+X),"O O";
v 110 PRINT CHR(19);:FOR I=1
      TO Y:PRINT:NEXT I:
      PRINT TAB(X);"O O";
  120 NEXT G
b 130 PRINT CHR$(30);
s 130 PRINT AT 0,0;
d 130 PRINT @0,;
v 130 PRINT CHR$(19);
```

```
140 FOR I=1 TO 300 : GOSUB 3000
150 IF I$<>"" THEN LET I=9999
160 NEXT I
170 LET K=VAL("0"+I$)
180 IF K<N THEN PRINT
    "THERE ARE ";N-K;" GHOSTS LEFT"
190 IF K>N THEN PRINT
    "YOU CREATED ";K-N;" GHOSTS"
200 LET T=T+ABS(N-K)
210 FOR I=1 TO 1000 : NEXT I
220 NEXT R
230 PRINT "THERE ARE ";T;" GHOSTS"
240 PRINT "LEFT IN THE HOUSE."
250 IF T=0 THEN PRINT "WELL DONE!"
    : GOTO 4000
260 PRINT "YOU'LL JUST HAVE TO TRY AGAIN!"
270 FOR I=1 TO 1000 : NEXT I
280 GOTO 40
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

Lines 10-30 clear the screen and print the title of the game.

Line 40 sets the count of the number of ghosts that are not banished forever to zero (in T).

Line 50 is the start of the loop giving you ten rooms.

Lines 60-70 clear the screen and display the current room that you are in.

Lines 80-120 choose a random number of ghosts for this room (between one and eight), and display this number of ghost's eyes on the screen at random positions.

Line 130 sets the cursor to the top left hand corner of the screen again.

Lines 140-170 give you a certain time to press a key, and, if it is a number key, put its value into K.

Lines 180-190 display messages if you got the number wrong (either too small or too large).

Line 200 add the number of ghosts left in the room to the total count of ghosts left in T.

Lines 210-220 delay a while, and then repeat the loop for the next room.

Lines 230-240 display a message telling you how many ghosts were left in the house after you have gone through all ten rooms.

Line 250 check if you banished all the ghosts, and if so diplays a suitable message and stops the program.

Lines 260-280 tell you that you'll have to try again, since you didn't get all the ghosts, delays a while, and restarts the program from line 40.

Changes you can make:

1) You can change the speed of the game by changing the number in line 140. If you make the number smaller, you will not get as much time to try to count the number of ghosts. If you make it larger, you will get more time.

2) You could give each of the ten rooms a name instead of just a number. To do this, put each room name, in quotation marks, in DATA statements at the end of the program, such as

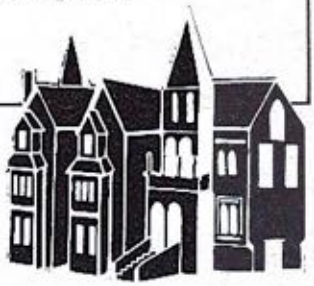
```
500 DATA "KITCHEN","BATHROOM","LIVING ROOM"
510 DATA "MAIN BEDROOM","GUEST ROOM"
etc.
```

Then, instead of line 70, which prints the room number on the screen, put a line that READs in the next room name, and prints it on the screen, for example,

```
70 READ R$ : PRINT "YOU ARE IN THE ";R$
```

Finally, you need a RESTORE instruction at the beginning of the game, so that if it is re-run, the DATA can be read again:

```
45 RESTORE
```



12: ESCAPE

You have been taken to a high-security prison for crimes you did not commit. You are determined to get out and put the real culprit behind bars. You plan to escape by digging a tunnel out of the prison. A fellow prisoner draws you a map of the prison and its surroundings. From where your cell is positioned you must decide in what direction you want to dig your tunnel and how long it should be. Your cell is marked by a # sign. If you dig too far you will collapse from exhaustion and be discovered. If your tunnel is too short you will come up inside the prison. And of course there may be problems when you get out....

```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "ESCAPE"
 30 PRINT
 40 LET L=FNR(40)*15
 50 LET X=FNR(4)
 60 LET Y=FNR(6)
 70 PRINT "THE ENCLOSURE IS ";L;" FEET SQUARE"
 80 FOR I=1 TO 2000 : NEXT I
 90 GOSUB 2000 : GOSUB 700
100 GOSUB 280 : GOSUB 260
110 PRINT "HOW FAR TO DIG";
120 INPUT F
130 GOSUB 260 : GOSUB 900
140 GOSUB 2000
150 IF F<(L/100)*(70+FNR(20)) THEN GOTO 170
160 PRINT "YOU COLLAPSED FROM EXHAUSTION" :
    GOTO 4000
170 LET X1=X/5*L : LET Y1=Y/7*L
180 LET X1=X1+X9*F : LET Y1=Y1+Y9*F
190 IF Y1<0 THEN GOTO 300
200 IF Y1>L THEN GOTO 400
210 IF X1<0 THEN GOTO 500
220 IF X1>L THEN GOTO 600
230 PRINT "YOU CAME UP INSIDE THE ENCLOSURE"
240 PRINT "BAD LUCK!"
250 GOTO 4000
```

```
b 260 PRINT CHR$(30);
s 260 PRINT AT 0,0;
d 260 PRINT @(0),;
v 260 PRINT CHR$(19);
  270 RETURN
b 280 PRINT TAB(X*2+4,Y+4);"#";
s 280 PRINT AT Y+4,X*2+4;"#";
d 280 PRINT @(Y*32+X*2+100),"#";
v 280 PRINT CHR$(19);:FOR Q=1 TO Y+4:PRINT:NEXT
    Q:PRINT TAB(X*2+4);"#";
  290 RETURN
  300 PRINT "YOU ARE IN THE FOREST"
  310 LET X=FNR(3)-2 : LET Y=FNR(3)-2
  320 LET A$="COMPLETELY" : FOR I=1 TO 6
  330 PRINT "YOU ARE ";A$;" LOST..."
  340 GOSUB 900 : LET X=X+X9 : LET Y=Y+Y9
  350 IF X=0 AND Y=0 THEN GOTO 390
  360 LET A$="VERY"
  370 IF X=0 OR Y=0 THEN LET A$="SLIGHTLY"
  380 NEXT I : PRINT "THE DOGS FOUND YOU!
    BAD LUCK" : GOTO 4000
  390 PRINT "YOU ESCAPED!!!!! WOW!!" :GOTO 4000
  400 PRINT "YOU ARE BY A ROAD"
  410 IF X=1 OR X=4 OR FNR(10)>6 THEN GOTO 450
  420 PRINT "THE GATE HOUSE COLLAPSED"
  430 PRINT "ON TOP OF YOU! BAD LUCK!"
  440 GOTO 4000
  450 LET A$="CAR"
  460 GOTO 620
  500 PRINT "YOU ARE IN A MINEFIELD"
  510 PRINT "THERE ARE MINES ALL AROUND"
  520 GOTO 310
  600 PRINT "YOU ARE BY A RAILWAY"
  610 LET A$="TRAIN"
  620 FOR I=1 TO FNR(600)+200 : NEXT I
  630 LET C=FNR(26) : GOSUB 2000
  640 PRINT "PRESS THE ";CHR$(64+C);" KEY TO
    CATCH A ";A$
```

```
vbd 650 FOR I=1 TO 150
s   650 FOR I=1 TO 50
    660 GOSUB 3000 : IF I$="" THEN NEXT I
    670 IF I$=CHR$(64+C) OR I$=CHR$(96+C) THEN
        GOTO 690
    680 PRINT "YOU HAVE BEEN SPOTTED. BAD LUCK!" :
        GOTO 4000
    690 PRINT "YOU ESCAPED!!!!! PHEW!!" :
        GOTO 4000
vbs 700 PRINT : PRINT
d   700 PRINT
    710 PRINT TAB(5);"* *  FOREST  *"
    720 PRINT TAB(7);"*"
    730 PRINT TAB(4);"I--------I  I-I"
    740 PRINT "O O I         I  I-I R"
    750 PRINT " O  I         I  I-I A"
    760 PRINT "M   I         I  I-I I"
    770 PRINT "I O I         I  I-I L"
    780 PRINT "N O I         I  I-I W"
    790 PRINT "E   I         I  I-I A"
    800 PRINT "S   I I----I I  I-I Y"
    810 PRINT "  O I-IGATEI-I  I-I"
    820 PRINT TAB(16);"I-I"
    830 PRINT "----------------------"
    840 PRINT " -  -  - ROAD -  -  -"
    850 PRINT "----------------------";
    860 RETURN
    900 PRINT "DIRECTION (N,S,E,W)"
    910 LET X9=0 : LET Y9=0
    920 INPUT D$
    930 IF D$="N" OR D$="n" THEN LET Y9=-1
    940 IF D$="S" OR D$="s" THEN LET Y9=1
    950 IF D$="W" OR D$="w" THEN LET X9=-1
    960 IF D$="E" OR D$="e" THEN LET X9=1
    970 IF X9=0 AND Y9=0 THEN GOTO 920
    980 RETURN
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

Changes you can make:

1) If your computer has a clock, you could incorporate this in the program to tell you how long you took to escape. Read your computer manual about using the clock, and then add lines so that the time is set to zero right at the start of the program, and is then printed out at the end.

2) You can change the speed of the reaction testing part of the program that is used if your tunnel is near the road or the railway. To make the game more difficult, change the number in line 650 to a smaller number.

13: SPACE DOGFIGHT

On the edge of the great space battle between your fleet and the space fighters of Xaaxon, you have just engaged an enemy fighter in a space dogfight above a deserted moon. You creep up behind the ship, and have to try to decide what action to take at each point in the game.

While the game is running, press the letter keys that correspond to the actions you want to take. Note also that at most times you can press 'F' for fire, even when, for instance, you are overtaking the enemy ship.

```
 10 GOSUB 1000 : GOSUB 2000
 20 PRINT "SPACE DOGFIGHT"
 30 PRINT
 40 LET D=5
 50 LET H=5
 60 PRINT "PRESS A KEY TO START";
 70 GOSUB 3000 : IF I$="" THEN GOTO 70
 80 FOR I=1 TO 1500 : NEXT I : GOSUB 2000
 90 IF D>0 THEN PRINT "YOU ARE BEHIND HIM"
100 IF D<0 THEN PRINT "HE IS BEHIND YOU"
110 IF D=0 THEN PRINT "YOU ARE LEVEL"
120 FOR I=1 TO 100 : GOSUB 3000 : IF I$="
    " THEN NEXT I
130 IF I$<>"F" AND I$<>"f" THEN GOTO 200
140 IF D=0 AND FNR(10)<3 THEN PRINT "YOU HIT
    HIM!" : GOTO 4000
150 IF D>0 AND D<3 AND FNR(10)<3 THEN PRINT
    "HE LOST SPEED" : LET D=D-1
160 IF D>2 THEN PRINT "TOO FAR AWAY"
170 IF D<0 THEN PRINT "YOU CAN'T SHOOT BEHIND"
200 PRINT "YOU CAN:     E-CLIMB"
210 PRINT "             A-ACCEL."
220 PRINT "             D-DECEL."
230 PRINT "             X-DIVE"
```

```
250 GOSUB 3000: IF I$="" THEN GOTO 250
260 IF I$="A" OR I$="a" THEN LET D=D-1 :
    GOTO 800
270 IF I$="D" OR I$="d" THEN LET D=D+1 :
    GOTO 800
280 IF I$="X" OR I$="x" THEN GOTO 400
290 IF I$="E" OR I$="e" THEN GOTO 500
300 IF ABS(D)>8 THEN PRINT "YOU HAVE LOST SIGHT
    OF HIM." : GOTO 4000
310 GOTO 250
400 GOSUB 2000
410 PRINT "YOU ARE DIVING...."
420 LET H=H-1
425 IF H=0 THEN PRINT "TOO LOW!" : GOTO 480
430 PRINT "YOU CAN:     E-PULL OUT"
440 PRINT "             X-LOOP UNDER"
450 FOR I=1 TO 200 : GOSUB 3000 : IF I$="" THEN
    NEXT I
460 IF I$="E" OR I$="e" THEN LET D=D-2 :
    GOTO 800
470 IF (I$="X" OR I$="x") AND FNR(10)<6 THEN
    LET D=D+2 : GOTO 800
480 PRINT "YOU CRASHED INTO THE MOON!!!"
490 GOTO 4000
500 GOSUB 2000
510 PRINT "YOU ARE CLIMBING...."
520 LET H=H+1
```

```
525 IF H=8 THEN PRINT "TOO HIGH!" : GOTO 580
530 PRINT "YOU CAN:      E-LOOP OVER"
540 PRINT "              X-PULL OUT"
550 FOR I=1 TO 200 : GOSUB 3000 : IF I$="" THEN
    NEXT I
560 IF I$="X" OR I$="x" THEN LET D=D-2 : GOTO
    800
570 IF (I$="E" OR I$="e") AND FNR(10)<8
    THEN LET D=D+2 : GOTO 800
580 PRINT "YOU LOST THE ATMOSPHERE!!"
590 GOTO 4000
800 FOR I=1 TO 100 : GOSUB 3000 : IF I$="" THEN
    NEXT I
810 IF (I$="F" OR I$="f") AND (D=0 OR D=1) THEN
    GOTO 4000
820 IF D>0 OR FNR(10)<5 THEN GOTO 860
830 PRINT "HE FIRES...."
840 IF FNR(10)<4 THEN PRINT "AND HITS YOU!!!!!" :
    PRINT "HE'S DESTROYED!" : GOTO 4000
850 PRINT "...BUT MISSES."
860 IF D<0 OR D>3 THEN GOTO 900
870 IF FNR(10)<4 THEN PRINT "HE DIVES AND LOOPS
    UNDER" : LET D=-D
900 GOTO 80
```

For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

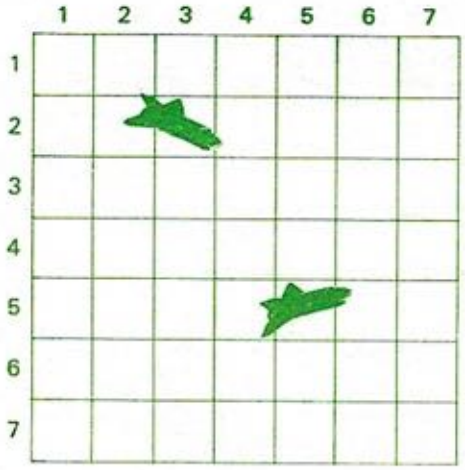
ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

14: DEATH RAID

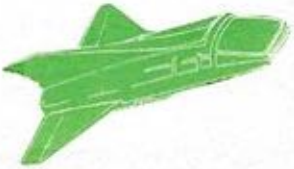
You are flying towards the distant location of the largest space war in history. Your side is winning and you are going to support them. Suddenly, fleeing enemy ships appear on your front scanner, as they come out of hyperspace to refuel. You must try to locate their position and speedily use your laser guns to shoot them down before they can return to the safety of hyperspace.

The ships appear on a 7 by 7 grid on your screen. You must type in the horizontal followed by the vertical position on the grid for your laser to fire at. Just type in the two digits — there is no need to press the 'return' or 'enter' key. Type as quickly as you can before the enemy has a chance to get away.




```
10 GOSUB 1000 : GOSUB 2000
20 PRINT "DEATH RAID"
30 PRINT
40 LET S=0
50 FOR Y=1 TO 7
60 LET L$="---" : GOSUB 300
70 LET L$="I  " : GOSUB 300
80 NEXT Y
90 LET L$="---" : GOSUB 300
100 FOR G=1 TO 10
110 LET X=FNR(7)
```

```
   120 LET Y=FNR(7)
   130 LET P$="S" : GOSUB 400
   140 LET J$=""
   150 FOR I=1 TO 300
   160 GOSUB 3000
   170 LET J$=J$+I$
s  175 FOR K=1 TO 20 : NEXT K
   180 IF LEN(J$)<>2 THEN NEXT I : GOTO 230
vbd190 IF VAL(LEFT$(J$,1))<>X OR VAL(RIGHT$
       (J$,1))<>Y THEN GOTO 230
s  190 IF VAL(J$,1)<>X OR VAL(J$,2)<>Y
       THEN GOTO 230
   200 LET P$="*" : GOSUB 400
   210 LET S=S+1
   220 FOR I=1 TO 500 : NEXT I
   230 LET P$=" " : GOSUB 400
   240 NEXT G
   250 GOSUB 2000
   260 PRINT "YOU HIT ";S;" SPACESHIPS"
   270 GOTO 4000
   300 FOR X=1 TO 7
bds310 PRINT L$;
v  310 PRINT LEFT$(L$,2);
   320 NEXT X
vbd330 PRINT LEFT$(L$,1)
s  330 PRINT L$(1)
   340 RETURN
b  400 PRINT TAB(X*3-2,Y*2);P$P$;
s  400 PRINT AT Y*2+1,X*3-2;P$;P$;
d  400 PRINT @(Y*64+X*3-34),P$;P$;
v  400 PRINT CHR$(19);:FOR Q=1 TO Y*2+1:PRINT:
       NEXT Q:PRINT TAB(X*2-1);P$;
   410 RETURN
```



For the different computers

BBC/Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

How the program works:

Lines 10-30 clear the screen and display the title of the gane.

Line 40 sets your score to zero (in S).

Lines 50-90 display a grid on the screen, using the subroutine at lines 300-340.

Line 100 is the start of a loop giving you ten ships to hit.

Lines 110-120 choose the random position of the enemy ship on your grid.

Line 130 marks this position with an 'S'. The subroutine at 400 displays the string at X,Y on the grid.

Line 140 makes the input string empty.

Line 150 is the start of the loop giving you a certain time to type in the two numbers.

Line 160 gets a character from the keyboard (if there) and adds it to the end of the string (if nothing is pressed, then nothing will be added).

Line 170 delays a while.

Line 180 checks if you have pressed two keys, and if not, repeats the loop, and then goes on to line 230.

Line 190 checks if you got the two numbers correct. If not, the computer will go to line 230.

Line 200 displays a '*' where the ship was, to tell you that you hit it.

Line 210 increases your score by one.

Line 220 delays a while to let you see that you hit the ship

Line 230 clears the ship to a space on the screen.

Line 240 is the end of the loop, giving you another go, unless you have had ten.

Lines 250-270 clear the screen, tell you how many ships you hit, and stop the program.

Lines 300-340 are a subroutine that prints a repeated row of the string L$, ending with the single character that begins L$ (to make the picture tidy).

Lines 400-410 print the string P$ at the correct position on the screen, depending on X and Y. For some computers, P$ is printed twice to make it easier to see.

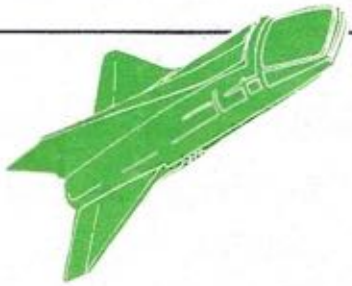
Lines 1000-4000 contain the standard routines.

Changes you can make:

1) You can change the number in line 100 to give you a greater or smaller number of enemy ships. If you change the ten to twelve,you will get twelve ships attacking.

2) You can make the game more of less difficult by changing the value in line 150. This controls the length of time allowed for you to type in the two digits. If you decrease the number to 150, the game will get faster, and it will be more difficult to fire at the enemy in time.

3) See if you can work out how to change the scoring system, so that you get a different score depending on how long you took to shoot the enemy ship. You will need to change line 210, and also the message in line 260, since the score will no longer be a count of the number of ships hit. Hint - use the value of the variable in the loop that gives you a certain time to enter the two digits.



ZX-Spectrum

A whole village depends on you. You, Wild Bill, are the only person left who can defend the village from a group of advancing Indians. You have got your trusty six-shooter, and you use it to kill the Indians hiding behind the cacti, shooting one bullet at a time. But you have to keep pausing to reload every six shots, in which time the enemy will creep forwards and dead Indians will be replaced with live ones.

The Indians will only come out of hiding for a few seconds — so be quick! If you do not shoot, they will once again come closer. Let them creep *too* close and you will be overwhelmed.

To move up and down press the A and Z keys respectively. To fire, press the RETURN or ENTER key.

```
 10 REM COWBOYS AND INDIANS
 20 REM FOR THE ZX-SPECTRUM
 50 GOSUB 1000
 60 GOSUB 1100
 70 GOSUB 1300
 80 GOSUB 1500
 90 GOSUB 1600
100 GOSUB 2700
110 IF K=1 THEN LET S=S+1:LET K=0:GOSUB 2700
120 IF CM=0 THEN GOTO 160
130 LET C=7:GOSUB 1700
140 LET CY=CY+CM : LET CM=0
150 LET C=5:GOSUB 1700
160 IF BE=0 THEN GOTO 270
170 LET C=7:GOSUB 1900
180 LET BX=BX+1
190 IF BX>30 THEN LET BE=0:GOTO 270
200 LET P=ATTR(BY,BX)-56
210 IF P=7 THEN LET C=2:
    GOSUB 1900:GOTO 270
220 IF P=4 THEN GOSUB
    2000:LET BE=0:
    GOTO 270
```

```
230 FOR J=1 TO 3
240 IF BY>Y(J)-3 AND BY<Y(J)+3 THEN LET I(J)=0:
    LET M(J)=1: LET K=1
250 NEXT J
260 LET BE=0
270 FOR J=1 TO 3
280 IF M(J)=0 THEN GOTO 340
290 LET C=7:GOSUB 1800
300 LET H(J)=1-H(J):LET M(J)=0
310 IF I(J)=0 THEN GOSUB 2400:GOTO 340
320 GOSUB 2300
330 LET C=5:GOSUB 1800
340 NEXT J
350 LET I$=INKEY$
370 IF I$="" AND BE=0 THEN LET Z=Z+1:GOTO 420
380 LET Z=0
390 IF (I$="A" OR I$="a") AND CR=0 AND CY>1
    THEN LET CM=-1
400 IF (I$="Z" OR I$="z")AND CR=0 AND CY<18
    THEN LET CM=1
410 IF I$=CHR$(13) AND CR=0 AND BE=0
    THEN GOSUB 2500
420 IF Z=40 THEN GOTO 440
430 LET CR=CR+SGN(CR) : IF CR<20 THEN GOTO 460
440 LET MX=MX-5:IF MX=0 THEN GOSUB 2800:STOP
450 GOTO 80
460 FOR J=1 TO 3
470 IF I(J)=1 AND H(J)=0 AND RND<.01
    THEN LET M(J)=1
480 NEXT J
490 FOR J=1 TO 3
500 IF H(J)=0 OR I(J)=0 THEN GOTO 520
510 LET I(J)=I(J)+1:IF I(J)=MX+10
    THEN LET M(J)=1:LET I(J)=1
520 NEXT J
530 IF I(1)+I(2)+I(3)=0 THEN GOSUB 2600:STOP
540 GOTO 110
1000 REM INITIALIZE
1020 LET MX=25:LET S=0:LET BN=0
```

```
1030 DIM Y(3):DIM I(3):DIM H(3):DIM M(3)
1050 RETURN
1100 REM DEFINE CHARACTERS
1110 FOR I=65 TO 82
1120 FOR J=0 TO 7
1130 READ D
1140 POKE USR CHR$(I)+J,D
1150 NEXT J : NEXT I
1160 RETURN
1200 REM DISPLAY TITLE
1210 CLS
1220 PRINT AT 4,14;INK 2;"COWBOYS"
1230 PRINT AT 6,16;INK 4;"AND"
1240 PRINT AT 8,14;INK 5;"INDIANS"
1250 INK 0: PRINT AT 16,0;"    A TO GO UP"
1260 PRINT "    Z TO GO DOWN"
1270 PRINT "ENTER TO FIRE"
1280 FOR I=1 TO 500 : NEXT I
1290 RETURN
1500 REM SET UP VARIABLES
1510 LET CX=2:LET CY=18:LET CR=0:LET CM=1
1520 LET BX=0:LET BY=0:LET BE=0
1530 LET Y(1)=3:LET Y(2)=10:LET Y(3)=17
1540 LET I(1)=1:LET I(2)=1:LET I(3)=1
1550 LET H(1)=1:LET H(2)=1:LET H(3)=1
1560 LET M(1)=1:LET M(2)=1:LET M(3)=1
1570 LET IX=4:LET K=0:LET Z=0
1580 RETURN
1600 REM DISPLAY GAME SCREEN
1610 INK 7:CLS
1620 LET C=1:GOSUB 2100
1630 LET C=2:GOSUB 2100
1640 LET C=3:GOSUB 2100
1650 RETURN
1700 REM DRAW COWBOY
1710 INK C
1720 PRINT AT CY,CX;CHR$(144)
1730 PRINT AT CY+1,CX;CHR$(145)
1740 PRINT AT CY+2,CX;CHR$(146)
```

```
1750 RETURN
1800 REM DRAW INDIAN
1810 INK C
1820 PRINT AT Y(J)-H(J)*2,MX+IX;CHR$(159)
1830 PRINT AT Y(J)-H(J)*2+1,MX+IX;CHR$(160)
1840 PRINT AT Y(J)-H(J)*2+2,MX+IX;CHR$(161)
1850 RETURN
1900 REM DRAW BULLET
1910 PRINT AT BY,BX;INK C;"-"
1920 RETURN
2000 REM SPLAT CACTUS
2010 INK 7
2020 PRINT AT BY-1,BX;" "
2030 PRINT AT BY,BX;" "
2040 PRINT AT BY+1,BX;" "
2050 RETURN
2100 REM DRAW CACTUS
2110 INK 4
2120 PRINT AT Y(C),MX;CHR$(147);CHR$(152);
     CHR$(155)
2130 PRINT AT Y(C)+1,MX;CHR$(148);CHR$(153);
     CHR$(156)
2140 PRINT AT Y(C)+2,MX;CHR$(149);CHR$(154);
     CHR$(157)
2150 PRINT AT Y(C)+3,MX;CHR$(150);CHR$(153);
     CHR$(158)
2160 PRINT AT Y(C)+4,MX;CHR$(151);CHR$(153);
     CHR$(158)
2170 RETURN
2200 REM BANG SOUND
2210 FOR I=1 TO 15 STEP 2:BEEP 0.01/I,0:
     BEEP 0.01/I,10:NEXT I
2220 RETURN
2300 REM PING SOUND
2310 BEEP .05,24
2320 RETURN
2400 REM DEAD INDIAN SOUND
2410 BEEP .2,-10
2420 RETURN
```

```
2500 REM GUN FIRE ROUTINE
2510 LET BE=1: LET BX=CX+1: LET BY=CY+1:
     LET BN=BN+1
2520 IF BN=7 THEN LET BN=0: LET BE=0:
     LET CR=1: RETURN
2530 GOSUB 2200
2540 RETURN
2600 REM PLAYER WON
2610 CLS:INK 0
2620 PRINT "THE REST OF THE INDIANS"
2630 PRINT "HAVE RUN AWAY. YOU SAVED"
2640 PRINT "THE VILLAGE!"
2650 RETURN
2700 REM DISPLAY SCORE
2710 PRINT AT 0,0;INK 0;"INDIANS KILLED: ";S
2720 RETURN
2800 REM PLAYER LOST
2810 CLS:INK 0
2820 PRINT "YOU HAVE BEEN OVERWHELMED"
2830 PRINT "BY THE INDIANS. BAD LUCK!"
2840 RETURN
3000 DATA 60,60,255,60,60,60,56,254
3010 DATA 146,146,146,146,186,154,138,138
3030 DATA 138,254,40,40,40,40,44,48
3050 DATA 0,0,0,0,0,4,14,14
3060 DATA 14,14,14,14,14,15,7,1
3070 DATA 0,0,8,28,28,28,28,28
3080 DATA 28,28,28,28,28,28,28,31
3090 DATA 15,7,0,0,0,0,0,0
3100 DATA 56,124,124,254,254,254,254,254
3110 DATA 254,254,254,254,254,254,254,254
3120 DATA 255,255,255,254,254,254,254,254
3130 DATA 0,0,0,0,0,0,0,32
3140 DATA 112,112,112,112,112,112,112,112
3150 DATA 240,224,192,0,0,0,0,0
3160 DATA 0,0,0,0,0,0,0,0
3200 DATA 4,8,8,60,60,60,60,28
3210 DATA 124,68,68,68,68,68,68,68
3220 DATA 124,40,40,40,40,40,104,24
```

VIC

A whole village depends on you. You, Wild Bill, are the only person left who can defend the village from a group of advancing Indians. You have got your trusty six-shooter, and you use it to kill the Indians hiding behind the cacti, shooting one bullet at a time. But you have to keep pausing to reload every six shots, in which time the enemy will creep forwards and dead Indians will be replaced with live ones.

The Indians will only come out of hiding for a few seconds — so be quick! If you do not shoot, they will once again come closer. Let them creep *too* close and you will be overwhelmed.

To move up and down press the A and Z keys respectively. To fire, press the RETURN or ENTER key.

```
 10 REM COWBOYS AND INDIANS
 20 REM FOR THE VIC
 40 GOSUB 1000
 60 GOSUB 1200
 80 GOSUB 1500
 90 GOSUB 1600
100 GOSUB 2700
110 IF K=1 THEN S=S+1:K=0:GOSUB 2700
120 IF CM=0 THEN 160
130 C=1:GOSUB 1700
140 CY=CY+CM : CM=0
150 C=3:GOSUB 1700
160 IF BE=0 THEN 270
170 C=1:GOSUB 1900
180 BX=BX+1
190 IF BX>21 THEN BE=0:GOTO 270
200 P=PEEK(38378+BY*22+BX) AND 7
210 IF P=1 THEN C=2:GOSUB 1900:GOTO 270
220 IF P=5 THEN GOSUB 2000:BE=0:GOTO 270
230 FOR J=1 TO 3
240 IF BY>Y(J)-4 AND BY<Y(J)+4
    THEN I(J)=0:M(J)=1:K=1
250 NEXT
```

```
260 BE=0
270 FOR J=1 TO 3
280 IF M(J)=0 THEN 340
290 C=1:GOSUB 1800
300 H(J)=1-H(J):M(J)=0
310 IF I(J)=0 THEN GOSUB 2400:GOTO 340
320 GOSUB 2300
330 C=3:GOSUB 1800
340 NEXT
350 GET I$
370 IF I$="" AND BE=0 THEN Z=Z+1:GOTO 420
380 Z=0
390 IF I$="A" AND CR=0 AND CY>2 THEN CM=-1
400 IF I$="Z" AND CR=0 AND CY<19 THEN CM=1
410 IF I$=CHR$(13) AND CR=0 AND BE=0
    THEN GOSUB 2500
420 IF Z=40 THEN 440
430 CR=CR+SGN(CR) : IF CR<20 THEN 460
440 MX=MX-3:IF MX=1 THEN GOSUB 2800:END
450 CR=0:GOTO 80
460 FOR J=1 TO 3
470 IF I(J)=1 AND H(J)=0 AND RND(1)<.01
    THEN M(J)=1
480 NEXT
490 FOR J=1 TO 3
500 IF H(J)=0 OR I(J)=0 THEN 520
510 I(J)=I(J)+1:IF I(J)=MX+10 THEN M(J)=1:
    I(J)=1
520 NEXT
530 IF I(1)+I(2)+I(3)=0 THEN GOSUB 2600:END
540 GOTO 110
1000 REM INITIALIZE
1010 MX=16:S=0:BN=0
1020 DIM Y(3),I(3),H(3),M(3)
1030 POKE 36879,25
1040 RETURN
1200 REM DISPLAY TITLE
1210 PRINT CHR$(147)
```

```
1220 PRINT TAB(8);CHR$(159);"COWBOYS"
1230 PRINT TAB(10);CHR$(30);"AND"
1240 PRINT TAB(8);CHR$(28);"INDIANS"
1250 PRINT:PRINT
1260 PRINT CHR$(144)
1270 PRINT "  A TO GO UP"
1280 PRINT "  Z TO GO DOWN"
1290 PRINT "RET TO FIRE"
1300 FOR I=1 TO 2000:NEXT
1310 RETURN
1500 REM SET UP VARIABLES
1510 CX=2:CY=18:CR=0:CM=1
1520 BX=0:BY=0:BE=0
1530 Y(1)=5:Y(2)=12:Y(3)=19
1540 I(1)=1:I(2)=1:I(3)=1
1550 H(1)=1:H(2)=1:H(3)=1
1560 M(1)=1:M(2)=1:M(3)=1
1570 IX=5:K=0:Z=0
1580 RETURN
1600 REM DISPLAY GAME SCREEN
1610 PRINT CHR$(147);
1620 C=1:GOSUB 2100
1630 C=2:GOSUB 2100
1640 C=3:GOSUB 2100
1650 RETURN
1700 REM DRAW COWBOY
1710 P=CX+CY*22+7658
1720 POKE P,254:POKE P+1,252:POKE P+22,225:
     POKE P+23,126
1730 POKE P+44,225:POKE P+45,97:POKE P+66,225:
     POKE P+67,126
1740 POKE P+88,225:POKE P+89,123
1750 FOR X=0 TO 1:FOR Y=0 TO 4:
     POKE P+X+Y*22+30720,C:NEXT:NEXT
1760 RETURN
1800 REM DRAW INDIAN
1810 P=MX+IX+(Y(J)-H(J)*3)*22+7658
1820 POKE P,85:POKE P+22,81:POKE P+44,91:
     POKE P+66,113
```

```
1830 FOR Y=0 TO 3:POKE P+Y*22+30720,C:NEXT
1840 RETURN
1900 REM DRAW BULLET
1910 P=BX+BY*22+7658
1920 POKE P,98:POKE P+30720,C
1930 RETURN
2000 REM SPLAT CACTUS
2010 P=BX+BY*22+7658+30720
2020 POKE P,1:POKE P+22,1:POKE P+44,1
2030 RETURN
2100 REM DRAW CACTUS
2110 P=MX+Y(C)*22+7658
2120 POKE P,225:POKE P+1,108:POKE P+2,252
2130 POKE P+22,225:POKE P+23,225:POKE P+24,224:
     POKE P+25,225
2140 POKE P+44,124:POKE P+45,251:POKE P+46,224:
     POKE P+47,254
2150 POKE P+67,225:POKE P+68,224
2160 FOR X=0 TO 3:FOR Y=0 TO 3:
     POKE P+X+Y*22+30720,5:NEXT:NEXT
2170 RETURN
2200 REM BANG SOUND
2210 POKE 36877,240
2220 FOR V=15 TO 1 STEP -1
2230 POKE 36878,V
2240 NEXT
2250 FOR Q=1 TO 80:NEXT
2260 POKE 36877,0
2270 RETURN
2300 REM PING SOUND
2310 POKE 36874,252
2320 FOR V=15 TO 1 STEP -1
2330 POKE 36878,V
2340 NEXT
2350 FOR Q=1 TO 80:NEXT
2360 POKE 36874,0
2370 RETURN
2400 REM DEAD INDIAN SOUND
2410 FOR V=12 TO 1 STEP -1
```

```
2420 POKE 36874,V*10+130
2430 POKE 36878,V
2440 NEXT
2450 POKE 36874,0
2460 RETURN
2500 REM GUN FIRE ROUTINE
2510 BE=1:BX=CX+2:BY=CY+1:BN=BN+1
2520 IF BN=7 THEN BN=0:BE=0:CR=1:RETURN
2530 GOSUB 2200
2540 RETURN
2600 REM PLAYER WON
2610 PRINT CHR$(147)
2620 PRINT "THE REST OF THE"
2630 PRINT "INDIANS HAVE RUN AWAY."
2640 PRINT "YOU SAVED THE VILLAGE"
2650 RETURN
2700 REM DISPLAY SCORE
2710 PRINT CHR$(19);"INDIANS KILLED: ";S
2720 RETURN
2800 REM PLAYER LOST
2810 PRINT CHR$(147)
2820 PRINT "YOU HAVE BEEN"
2830 PRINT "OVERWHELMED BY THE"
2840 PRINT "INDIANS. BAD LUCK!"
2850 RETURN
```

BBC/Electron

A whole village depends on you. You, Wild Bill, are the only person left who can defend the village from a group of advancing Indians. You have got your trusty six-shooter, and you use it to kill the Indians hiding behind the cacti, shooting one bullet at a time. But you have to keep pausing to reload every six shots, in which time the enemy will creep forwards and dead Indians will be replaced with live ones.

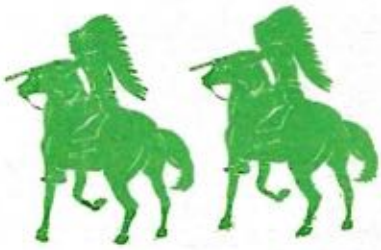
The Indians will only come out of hiding for a few seconds — so be quick! If you do not shoot, they will once again come closer. Let them creep *too* close and you will be overwhelmed.

To move up and down press the A and Z keys respectively. To fire, press the RETURN or ENTER key.

```
 10 REM COWBOYS AND INDIANS
 20 REM FOR THE BBC/ELECTRON
 30 MODE 5
 40 PROCINIT
 50 PROCCHARS
 60 PROCTITLE
 70 PROCENVELOPE
 80 PROCVARS
 90 PROCSCREEN
100 PROCSCORE
110 IF K=1 THEN S=S+1:K=0:PROCSCORE
120 IF CM=0 THEN 160
130 PROCCOWBOY(CX,CY,0,CL,0)
140 CY=CY+CM*20 : CM=0 : CL=1-CL
150 PROCCOWBOY(CX,CY,3,CL,0)
160 IF BE=0 THEN 270
170 PROCBULLET(BX,BY,0)
180 BX=BX+25
190 IF BX>1200 THEN BE=0 : GOTO 270
200 P=POINT(BX+10,BY)
210 IF P=0 THEN PROCBULLET(BX,BY,1) : GOTO 270
220 IF P=2 THEN PROCSPLAT(BX-20,BY+50) : BE=0 :
    GOTO 270
```

```
 230 FOR J=1 TO 3
 240 IF BY>Y(J)-90 AND BY<Y(J)+110
     THEN I(J)=0:M(J)=1:K=1
 250 NEXT
 260 BE=0
 270 FOR J=1 TO 3
 280 IF M(J)=0 THEN 340
 290 PROCINDIAN(MX+IX,Y(J)+H(J)*100,0)
 300 H(J)=1-H(J):M(J)=0
 310 IF I(J)=0 THEN PROCDEAD:GOTO 340
 320 PROCPING
 330 PROCINDIAN(MX+IX,Y(J)+H(J)*100,3)
 340 NEXT
 350 I$=INKEY$(1)
 360 *FX15,1
 370 IF I$="" AND BE=0 THEN Z=Z+1:GOTO 420
 380 Z=0
 390 IF I$="A" AND CR=0 AND CY<980 THEN CM=1
 400 IF I$="Z" AND CR=0 AND CY>100 THEN CM=-1
 410 IF I$=CHR$(13) AND CR=0 AND BE=0
     THEN PROCFIRE
 420 IF Z=40 THEN 440
 430 CR=CR+SGN(CR) : IF CR<20 THEN 460
 440 MX=MX-200: IF MX=0 THEN PROCLOST:END
 450 CR=0:GOTO 80
 460 FOR J=1 TO 3
 470 IF I(J)=1 AND H(J)=0 AND RND(1)<.01
     THEN M(J)=1
 480 NEXT
 490 FOR J=1 TO 3
 500 IF H(J)=0 OR I(J)=0 THEN 520
 510 I(J)=I(J)+1 : IF I(J)=MX/10
     THEN M(J)=1:I(J)=1
 520 NEXT
 530 IF I(1)+I(2)+I(3)=0 THEN PROCWON:END
 540 GOTO 110
1000 DEF PROCINIT
1010 *FX11,1
1020 *FX12,1
```

```
1030 MX=800:S=0:BN=0:DIM Y(3),I(3),H(3),M(3)
1040 VDU 19,0,7;0;19,2,2;0;19,3,6;0;
1050 ENDPROC
1100 DEF PROCCHARS
1110 FOR C=224 TO 246
1120 VDU 23,C
1130 FOR I=1 TO 8
1140 READ D : VDU D
1150 NEXT : NEXT
1160 ENDPROC
1200 DEF PROCTITLE
1210 VDU 5
1220 MOVE 448,800 : GCOL 0,1
1230 PRINT "COWBOYS"
1240 MOVE 576,700 : GCOL 0,2
1250 PRINT "AND"
1260 MOVE 448,600 : GCOL 0,3
1270 PRINT "INDIANS"
1280 MOVE 0,300
1290 PRINT "  A TO GO UP"
1300 PRINT "  Z TO GO DOWN"
1310 PRINT "RET TO FIRE"
1320 PROCCACTUS(100,900) : PROCCACTUS(900,200)
1330 I=INKEY(200)
1340 ENDPROC
1400 DEF PROCENVELOPE
1410 ENVELOPE 1,0,0,0,0,0,0,0,64,-5,-2,-20,
     126,60
1420 ENVELOPE 2,0,0,0,0,0,0,0,64,-5,-10,-1,
     126,60
1430 ENVELOPE 3,1,-5,0,0,30,0,0,64,-4,-1,-1,
     126,60
1440 ENDPROC
1500 DEF PROCVARS
1510 CX=20:CY=80:CR=0:CM=1:CL=0
1520 BX=0:BY=0:BE=0
1530 Y(1)=250:Y(2)=550:Y(3)=850
1540 I(1)=1:I(2)=1:I(3)=1
1550 H(1)=1:H(2)=1:H(3)=1
```



```
1560 M(1)=1:M(2)=1:M(3)=1
1570 IX=200:K=0:Z=0
1580 ENDPROC
1600 DEF PROCSCREEN
1610 CLS
1620 PROCCACTUS(MX,Y(1))
1630 PROCCACTUS(MX,Y(2))
1640 PROCCACTUS(MX,Y(3))
1650 ENDPROC
1700 DEF PROCCOWBOY(X,Y,C,L,F)
1710 MOVE X,Y : GCOL 0,C
1720 VDU 224,8,10,225+F,8,10,227+L
1730 ENDPROC
1800 DEF PROCINDIAN(X,Y,C)
1810 MOVE X,Y : GCOL 0,C
1820 VDU 244,8,10,245,8,10,246
1830 ENDPROC
1900 DEF PROCBULLET(X,Y,C)
1910 MOVE X,Y : GCOL 0,C
1920 DRAW X+10,Y
1930 ENDPROC
2000 DEF PROCSPLAT(X,Y)
2010 MOVE X,Y : GCOL 0,0
2020 VDU 241,8,10,242,8,10,243
2030 ENDPROC
2100 DEF PROCCACTUS(X,Y)
2110 MOVE X,Y : GCOL 0,2
2120 VDU 229,234,237,10,8,8,8
2130 VDU 230,235,238,10,8,8,8
2140 VDU 231,236,239,10,8,8,8
2150 VDU 232,235,240,10,8,8,8
2160 VDU 233,235,240
```

```
2170 ENDPROC
2200 DEF PROCBANG
2210 SOUND &10,1,5,10
2220 ENDPROC
2300 DEF PROCPING
2310 SOUND &11,2,200,5
2320 ENDPROC
2400 DEF PROCDEAD
2410 SOUND &12,3,200,35
2420 ENDPROC
2500 DEF PROCFIRE
2510 BE=1:BX=CX+80:BY=CY-30:BN=BN+1
2520 IF BN=7 THEN BN=0:BE=0:CR=1:ENDPROC
2530 PROCCOWBOY(CX,CY,0,CL,0)
2540 PROCCOWBOY(CX,CY,3,CL,1)
2550 PROCBANG
2560 PROCCOWBOY(CX,CY,0,CL,1)
2570 PROCCOWBOY(CX,CY,3,CL,0)
2580 ENDPROC
2600 DEF PROCWON
2610 VDU 4,12 : COLOUR 3
2620 PRINT ''"THE REST OF THE"
2630 PRINT ''"INDIANS HAVE RUN"
2640 PRINT ''"AWAY. YOU SAVED"
2650 PRINT ''"THE VILLAGE."
2660 *FX12,0
2670 ENDPROC
2700 DEF PROCSCORE
2710 VDU 4,30
2720 PRINT "INDIANS KILLED: ";S
2730 VDU 5
2740 ENDPROC
```

```
2800 DEF PROCLOST
2810 VDU 4,12 : COLOUR 1
2820 PRINT ''"YOU HAVE BEEN"
2830 PRINT ''"OVERWHELMED BY"
2840 PRINT ''"THE INDIANS"
2850 SOUND 1,-10,16,8 : SOUND 1,-10,0,10
2860 *FX12,0
2870 ENDPROC
3000 DATA 60,60,255,60,60,60,56,254
3010 DATA 146,146,146,146,186,154,138,138
3020 DATA 130,159,130,130,130,130,130,130
3030 DATA 138,254,40,40,40,40,44,48
3040 DATA 138,254,40,40,40,40,56,12
3050 DATA 0,0,0,0,0,4,14,14
3060 DATA 14,14,14,14,14,15,7,1
3070 DATA 0,0,8,28,28,28,28,28
3080 DATA 28,28,28,28,28,28,28,31
3090 DATA 15,7,0,0,0,0,0,0
3100 DATA 56,124,124,254,254,254,254,254
3110 DATA 254,254,254,254,254,254,254,254
3120 DATA 255,255,255,254,254,254,254,254
3130 DATA 0,0,0,0,0,0,0,32
3140 DATA 112,112,112,112,112,112,112,112
3150 DATA 240,224,192,0,0,0,0,0
3160 DATA 0,0,0,0,0,0,0,0
3170 DATA 0,0,0,192,192,248,252,255
3180 DATA 254,255,255,254,254,255,255,255
3190 DATA 254,248,240,224,192,0,0,0
3200 DATA 4,8,8,60,60,60,60,28
3210 DATA 124,68,68,68,68,68,68,68
3220 DATA 124,40,40,40,40,40,104,24
```

Dragon

A whole village depends on you. You, Wild Bill, are the only person left who can defend the village from a group of advancing Indians. You have got your trusty six-shooter, and you use it to kill the Indians hiding behind the cacti, shooting one bullet at a time. But you have to keep pausing to reload every six shots, in which time the enemy will creep forwards and dead Indians will be replaced with live ones.

The Indians will only come out of hiding for a few seconds — so be quick! If you do not shoot, they will once again come closer. Let them creep *too* close and you will be overwhelmed.

To move up and down press the A and Z keys respectively. To fire, press the RETURN or ENTER key.

```
 10 REM COWBOYS AND INDIANS
 20 REM FOR THE DRAGON
 40 GOSUB 1000
 60 GOSUB 1200
 80 GOSUB 1500
 90 GOSUB 1600
100 GOSUB 2700
110 IF K=1 THEN S=S+1:K=0:GOSUB 2700
120 IF CM=0 THEN 160
130 C=2:GOSUB 1700
140 CY=CY+CM*5 : CM=0
150 C=3:GOSUB 1700
160 IF BE=0 THEN 270
170 C=2:GOSUB 1900
180 BX=BX+8
190 IF BX>240 THEN BE=0:GOTO 270
200 P=PPOINT(BX+6,BY)
210 IF P=2 THEN C=4:GOSUB 1900:GOTO 270
220 IF P=1 THEN GOSUB 2000:BE=0:GOTO 270
230 FOR J=1 TO 3
240 IF BY>Y(J)-26 AND BY<Y(J)+26 THEN I(J)=0:
    M(J)=1:K=1
250 NEXT
```

```
260 BE=0
270 FOR J=1 TO 3
280 IF M(J)=0 THEN 340
290 C=2:GOSUB 1800
300 H(J)=1-H(J):M(J)=0
310 IF I(J)=0 THEN GOSUB 2400:GOTO 340
320 GOSUB 2300
330 C=3:GOSUB 1800
340 NEXT
350 I$=INKEY$
370 IF I$="" AND BE=0 THEN Z=Z+1:GOTO 420
380 Z=0
390 IF I$="A" AND CR=0 AND CY>5 THEN CM=-1
400 IF I$="Z" AND CR=0 AND CY<160 THEN CM=1
410 IF I$=CHR$(13) AND CR=0 AND BE=0
    THEN GOSUB 2500
420 IF Z=40 THEN 440
430 CR=CR+SGN(CR) : IF CR<20 THEN 460
440 MX=MX-50:IF MX=0 THEN GOSUB 2800:END
450 CR=0:GOTO 80
460 FOR J=1 TO 3
470 IF I(J)=1 AND H(J)=0 AND RND(0)<.01
    THEN M(J)=1
480 NEXT
490 FOR J=1 TO 3
500 IF H(J)=0 OR I(J)=0 THEN 520
510 I(J)=I(J)+1:IF I(J)=MX/5 THEN M(J)=1:I(J)=1
520 NEXT
530 IF I(1)+I(2)+I(3)=0 THEN GOSUB 2600:END
540 GOTO 110
1000 REM INITIALIZE
1010 MX=200:S=0:BN=0
1020 DIM Y(3),I(3),H(3),M(3)
1030 RETURN
1200 REM DISPLAY TITLE
1210 CLS0
1220 PRINT @77,"COWBOYS";
1230 PRINT @143,"AND";
1240 PRINT @205,"INDIANS";
```

```
1250 PRINT @384,"    A TO GO UP  ";
1260 PRINT @416,"    Z TO GO DOWN";
1270 PRINT @448,"ENTER TO FIRE   ";
1280 FOR I=1 TO 2000:NEXT
1290 RETURN
1500 REM SET UP VARIABLES
1510 CX=16:CY=155:CR=0:CM=1
1520 BX=0:BY=0:BE=0
1530 Y(1)=40:Y(2)=100:Y(3)=160
1540 I(1)=1:I(2)=1:I(3)=1
1550 H(1)=1:H(2)=1:H(3)=1
1560 M(1)=1:M(2)=1:M(3)=1
1570 IX=30:K=0:Z=0
1580 RETURN
1600 REM DISPLAY GAME SCREEN
1610 PMODE 3,1:SCREEN 1,0
1620 COLOR 1,2:PCLS
1630 C=1:GOSUB 2100
1640 C=2:GOSUB 2100
1650 C=3:GOSUB 2100
1660 RETURN
1700 REM DRAW COWBOY
1710 DRAW "BM"+STR$(CX)+","+STR$(CY)
1720 COLOR C
1730 DRAW "D2L4R4D6L2D10R10U10L4U2R2U4R4L4U2L6"
```

```
1740 DRAW "BM+0,+18D10R2BM+2,-10D12R2"
1750 PAINT (CX+4,CY+4)
1760 RETURN
1800 REM DRAW INDIAN
1810 DRAW "BM"+STR$(MX+IX)+",
     "+STR$(Y(J)-H(J)*25)
1820 COLOR C
1830 DRAW "D6R2D2L4D10R10U18L8R4U2E2"
1840 DRAW "BM+0,+22D8L2BM-2,-8D10L2"
1850 PAINT (MX+IX+4,Y(J)-H(J)*25+4)
1860 RETURN
1900 REM DRAW BULLET
1910 COLOR C
1920 LINE (BX,BY)-(BX+5,BY),PSET
1930 RETURN
2000 REM SPLAT CACTUS
2010 DRAW "BM"+STR$(BX)+","+STR$(BY-8)+"C2"
2020 DRAW "NR1D1NR2D1NR3D1NR6D1NR7D1NR9D1NR9D1
     NR8D1NR9D1NR9D1"
2030 DRAW "NR8D1NR8D1NR7D1NR8D1NR9D1NR7D1NR5D1
     NR3D1NR2D1NR1"
2040 RETURN
2100 REM DRAW CACTUS
2110 DRAW "BM"+STR$(MX)+","+STR$(Y(C))
2120 DRAW "D16L2H2U8H2G2D12F4R4D8R10U10"
2130 DRAW "R4E4U16H2G2D12G2L2U14H4L2G4"
2140 PAINT (MX+4,Y(C)+4)
2150 RETURN
2200 REM BANG SOUND
```

```
2210 PLAY "O1T100":FOR Q=31 TO 7 STEP -2:PLAY
     "V"+STR$(Q)+"GB"
2220 NEXT
2230 RETURN
2300 REM PING SOUND
2310 PLAY "O4T50":FOR Q=31 TO 5 STEP -3:PLAY
     "V"+STR$(Q)+"B"
2320 NEXT
2330 RETURN
2400 REM DEAD INDIAN SOUND
2410 PLAY "T804V30CGO5C"
2420 RETURN
2500 REM GUN FIRE ROUTINE
2510 BE=1:BX=CX+10:BY=CY+10:BN=BN+1
2520 IF BN=7 THEN BN=0:BE=0:CR=1:RETURN
2530 GOSUB 2200
2540 RETURN
2600 REM PLAYER WON
2610 PRINT @384,"THE REST OF THE INDIANS HAVE"
2620 PRINT "RUN AWAY. YOU SAVED THE VILLAGE!"
2640 RETURN
2700 REM DISPLAY SCORE
2710 SCREEN 0,0:CLS 4
2720 PRINT @262,"INDIANS KILLED: ";S;
2730 FOR Q=1 TO 1000:NEXT
2740 SCREEN 1,0
2750 RETURN
2800 REM PLAYER LOST
2810 PRINT @384,"YOU HAVE BEEN OVERWHELMED"
2820 PRINT "BY THE INDIANS. BAD LUCK!"
2830 RETURN
```

HOW TO WRITE YOUR OWN PROGRAM: SABOTAGE

Sabotage is given as an example of how to write a simple game program. The first thing you need to do before writing a game is to think of an idea. The idea must really be based on what you know the computer can and cannot do. You should know, for example, how the computer can store and recall numbers and words, and how it can make decisions depending on these numbers or words. You must try to think through your idea in a logical way, and try to write down something like this:

1. There is a bomb on a spaceship.
2. There are eight places where it could be, but you only have time to check three of them.
3. If the bomb is found, you then have to defuse it quickly.
4. To defuse the bomb, you must guess a secret number. You have a limited number of guesses, but the computer tells you if your guess is too high or too low.

Once you have an idea of some sort you can start to plan out the game. If your idea isn't completely clear, then don't despair — you may find that it gets better as you go along. The best way to start is to write down a series of simple steps that make up the game. Write them in English, but try to split them up into things that you know the computer can do. In our case, the list would look something like:

1. Set up the game:
   (a) Start the game and print out the title.
   (b) Decide where the bomb is hidden and what the secret code is going to be.
2. Get a choice from the player:
   (a) Ask which section of the spaceship he wants to look in.
   (b) Get his answer.
   (c) Check if he guessed the right section.
3. Give the player three chances to find it. Make a loop that starts just before 2(a) and ends just after 2(c). This loop should give the player three chances to check a section of the spaceship.

4. If the player doesn't find the bomb after three tries, print a message saying that the ship has blown up, and stop the game.
5. If the player guesses the position of the bomb in time, print a message saying that he has found it.
6. Defusing the bomb:
   (a) Print a question asking for the code number to defuse the bomb.
   (b) Get an answer from the player.
   (c) Check if he got the correct code.
   (d) If not, print a message telling him if his guess was higher or lower than the actual code.
7. Give the player six chances to guess the code. Make a loop that starts just before 6(a) and finishes just after 6(d). This loop should give the player six guesses at the code to defuse the bomb.
8. If the player doesn't guess the code after six tries, print a message saying that the ship has blown up, and stop the game.
9. If the player guesses the code, print a message telling him so, print 'WELL DONE', and stop the game.

It may take several tries to get a plan like this. You must practise the process of converting an idea into very simple steps that you know the computer will be able to understand. If you are not sure whether the computer can do something, see if you can split that thing into even smaller stages.

Once you have a plan like the one above, the best thing to do is to try to convert each step into a few lines of BASIC. Try to do this on paper first. You will find that you are constantly moving around the program and changing things, and this is much easier on paper than on the computer itself. You might find it easier to go through the steps and change them into a mixture of BASIC and English. For example, you might write

```
Loop 10 times
 ..                    (you would put the part of the
 ..                    program that you want inside the
 ..                    loop here)
End of loop
```

You can convert this into a FOR. . .NEXT loop at a later stage. You should try to foresee the things that could happen during the game, and try to cater for them all. For instance you must make sure that, when there is an input from the player, your program checks that whatever he typed in is actually allowed by the game and is not too big or too small.

Here is a listing of a program that will play the game described above. Each section below corresponds to the step with the same number in the list above. Each section has been written separately, so the line numbers are not all in numerical order. This doesn't matter when you are typing a program into the computer, because the computer will sort the lines automatically into the right order.

Section 1

```
10 GOSUB 1000 : GOSUB 2000
20 PRINT "SABOTAGE"
30 PRINT
40 LET N=8 : LET P=FNR(N) : LET C=FNR(100)
```

Explanation

Line 10 sets up the random number generator and clears the screen. This is the same as in all the other programs in this book,

and is written like this to make it easier to change the programs to work on different computers.

Lines 20—30 print the title followed by a blank line.

Line 40 sets the number of sections in the spaceship to 8 (this number is placed in N). It chooses a random section for the bomb to be hidden in between 1 and N (which is 8), and puts this number into P. It also chooses a secret code number (between 1 and 100) to defuse the bomb, and stores this number in C.

Section 2

```
60 PRINT "WHICH SECTION TO LOOK IN (1-8)"
70 INPUT S
80 IF S<1 OR S>N THEN GOTO 70
90 IF S=P THEN GOTO 200
```

Explanation

Line 60 asks the player which section he wants to look in.

Line 70 gets his answer and puts it in S.

Line 80 checks if the number that he typed in is allowed. If it is less than 1, or more than the total number of sections in the spaceship, it is not allowed, so the program will go back and ask for another number.

Line 90 checks if the number that the player typed in is the same as the random number that the computer chose at the beginning of the program. If it is, then the player has found the bomb, so the computer will go to line 200 (see later).

Section 3

```
 50 FOR I=1 TO 3
100 NEXT I
```

Explanation

Line 50 is the start of the loop giving the player three chances to find the bomb. It is put just before line 60, which asks the player for his guess.

Line 100 is the end of this loop, and will make the computer go back to line 60 until the player has had three guesses.

Section 4

```
110 PRINT "THE SHIP HAS BLOWN UP!"
120 GOTO 4000
```

Explanation

Line 110 prints a message telling the player that the ship has blown up; he has had three guesses and has not found the bomb in time.

Line 120 then stops the program by going to line 4000, which has a STOP or END instruction in it (according to the type of computer).

Section 5

```
200 PRINT "YOU FOUND THE TIME BOMB!"
```

Explanation

Line 200 prints a message telling the player that he has found the time bomb.

Section 6

```
220 PRINT "GIVE A CODE TO DEFUSE IT"
230 INPUT C1
240 IF C1<1 OR C1>100 THEN GOTO 230
250 IF C1=C THEN GOTO 400
260 IF C1<C THEN PRINT "TOO LOW"
270 IF C1>C THEN PRINT "TOO HIGH"
```

Explanation

Line 220 prints a question asking the player for a code to defuse the bomb.

Line 230 gets an answer from the player and puts it in C1.

Line 240 makes sure that the answer is allowed — that is, between 1 and 100. If it is not, the computer will go back to line 230 and ask again.

Line 250 checks if the answer in C1 is the same as the random code chosen by the computer earlier. If it is, the computer will go to line 400.

Lines 260—270 print a message, either 'TOO HIGH' or 'TOO LOW', depending on the guess at the code.

Section 7

```
210 FOR I=1 TO 6
280 NEXT I
```

Explanation

Line 210 is the start of a loop, just before the question asking the player for a guess at the code. The loop will give the player six chances to guess the code.

Line 280 is the end of the loop, and will make the computer go back to line 220 until the player has had six guesses.

Section 8

```
290 PRINT "YOU TOOK TOO LONG"
300 PRINT "THE SHIP BLEW UP"
310 GOTO 4000
```

Explanation

Lines 290—300 come after the end of the loop giving the player six tries at the code. If by that time he has not guessed it, the computer will print a message to tell the player that he has taken too long and that the ship blew up.

Line 310 will stop the program.

Section 9

```
400 PRINT "WELL DONE. YOU GUESSED THE CODE"
410 GOTO 4000
```

Explanation

Line 400 prints a message if the player has guessed the code correctly (see line 250).

Line 410 stops the program.

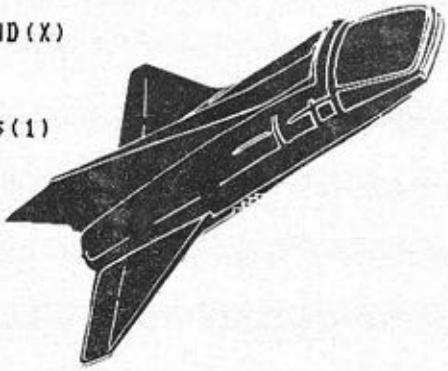
The last part of this program is the same as all the other programs in this book. You do not need to put a section like this at the end of your program, but it is useful if you want to make the game work on a different computer from your own.

BBC and Electron

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 END
```

Dragon

```
1000 DEF FNR(X)=RND(X)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$(1)
3010 RETURN
4000 END
```



VIC

```
1000 DEF FNR(X)
     =INT(RND(1)*X+1)
1010 RETURN
2000 PRINT CHR$(147);
     : RETURN
3000 GET I$
3010 RETURN
4000 END
```

ZX-Spectrum

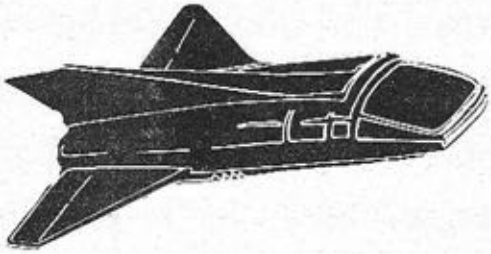
```
1000 DEF FNR(X)
     =INT(RND*X+1)
1010 RETURN
2000 CLS : RETURN
3000 LET I$=INKEY$
3010 RETURN
4000 STOP
```

Once you have written the BASIC program, type it into the computer, and SAVE it to tape or disc straight away. You should always do this, because you will probably change the program while you are testing it, and it is useful to have a copy of the original somewhere. Testing a program is a difficult and lengthy business.

How you do this testing depends on what sort of program it is. You should always try to get the computer to execute every section of the program, and try to make happen all the various things you have anticipated. For instance, when the program waits for an input from you, type in numbers or strings that you know are not allowed. If the program is written properly, the computer should notice that these are invalid, and should tell you or at least ask for the input again.

When the program is complete you are ready to write the scenario. For Sabotage, it could be something like this:

Someone has sabotaged your spaceship! You are just about to take off in your commander ship to save the empire, and you only have time to check three parts of your ship for the hidden time bomb. If you manage to find the bomb in time, you must use your expertise to defuse it while your ship is warming up for the flight — but be quick. . . .



BASIC is the name of the computer language that your machine can understand. It isn't like English, since you have to write everything very precisely and carefully. The smallest mistake would probably make the computer misunderstand you, and do something unexpected.

When you want to tell the computer to do something, you usually write a program. A program is a series of simple, logical steps that you want the computer to follow. Each step is written using words from the BASIC language, and you store them in the computer by giving them line numbers at the beginning of each instruction. Usually you use line numbers that go up in steps of ten, so that if you decide to put an extra line between two others, you can give it a line number that goes between the other two. Unless told otherwise, the computer goes through these lines one by one, starting at the first and working through in order of increasing line numbers.

To understand the programs in this book, you must remember that the computer is VERY stupid. In fact, it is so stupid that it needed a very good programmer to make it understand BASIC in the first place. So we have to ask it to do lots of simple things which, when put together, make it do what we want.

First, you must know how the computer remembers things. You use the word LET (see glossary) to tell it to put something into its memory. The place that holds this 'thing' is like a box. You give the box a name, and the computer will use that box whenever you use its name in a program. If the box holds a number it is called a 'numeric variable', and is given a single-letter name such as B. Some computers allow more complicated names, such as B1 or even Box. If the box holds a string of letters, digits or any other characters it is called a 'string variable'. As before it is given a name, followed by a dollar sign — hence N$ or P1$, for example.

So

```
LET Q=12
```

puts the number 12 into the numeric variable Q, whereas

```
LET N$="DANIEL"
```

puts the word DANIEL into the string variable N$. Notice that when you use string variables, the string itself has quotation marks before and after it.

You can also use LET to do maths on numbers and other variables. For example,

```
LET W=Q+5
```

will make the computer fetch the number stored in the variable Q, add five to it, and save the result in variable W. As well as +, you can use — for subtraction, * for multiplication, /for division and ^ for powers of a number. You can also use + with string variables. 'Adding' two strings means putting the second one on to the end of the first. For example:

```
LET A$="HELLO"
LET B$=" THERE"
LET C$=A$+B$
```

would make C$ hold the words HELLO THERE.

Doing things like this with variables is only useful if you can ask the computer to show you the results on the screen. The PRINT command is used for this, and can do two things. It can display messages on the screen, for example:

```
PRINT "HELLO THERE"
```

and it can be used to show you what numbers or strings are stored in variables, for example:

```
PRINT A
PRINT N$
```

You can combine these by putting more than one item in a PRINT command. To do this, you separate each item with a semicolon or comma. Usually a semicolon will display the items with no spaces between them, while a comma will space the items neatly across the screen.

Using sound on your computer

Adding sound effects to a computer game can make it even more fun. The computers that this book is written for can all make sounds of some sort, although the BBC is the best at it. Try the following examples on your computer, and try putting them in a program you have typed in, so that you get sound effects at the right time during the game.

Dragon

The simple command to make a sound is

```
SOUND P,D
```

where P and D are numbers or numeric variables that tell the computer what note you want to play, and for how long, respectively. P can be between 1 and 255; a value of 89 will cause the computer to play the note middle C. D can also be between 1 and 255: each unit is 1/16th of a second, so that a value of 8 will make the note last for half a second.

You can combine lots of SOUND commands to make more interesting noises. The best way to do this is to put them in a loop, and use the loop variable to change the note or the length of the note or even both. For example:

```
10 FOR I=20 TO 50
20 SOUND I,1
30 NEXT I
```

If you want to send everyone totally mad, try the following:

```
10 LET P=RND(255)
20 LET D=RND(4)
30 SOUND P,D
40 GOTO 10
```

The Dragon also has a more sophisticated way of playing music. The PLAY command lets you use note names instead of numbers, and has several powerful features. If you are interested, read about this command in the Dragon manual.

ZX-Spectrum

The ZX-Spectrum has a command BEEP, which produces a particular note for a chosen length of time:

```
BEEP D,P
```

You must use numbers or numeric variables for D and P. D is the length of the note, and each unit is one second. P is a number that gives you a particular note, and each unit is one semitone above middle C. (It can be negative. Thus 3 gives three semitones above, and —1 one semitone below, middle C.) Experiment with different BEEP instructions to see what it can do. For example:

```
BEEP 1,12
BEEP 0.5,-2
```

You can put the BEEP command into a loop to make more interesting sounds:

```
10 FOR I=0 TO 24
20 BEEP 0.05,I
30 NEXT I
```

or

```
10 FOR I=1 TO 25
20 BEEP I/500,12
30 BEEP I/500,19
40 NEXT I
```

BBC

The BBC computer has a very complicated system of producing sounds. In fact, you can have three notes all playing at once, together with 'white noise', which can sound a bit like a gun shot or explosion. The simplest way of making a noise is:

```
SOUND C,E,P,D
```

You must give a channel number in C. This tells the computer which of the four channels (three notes plus noise) to use. It can be either 0 (for noise) or 1 to 3 (for notes). E should be a minus number,

which tells the computer how loud to play the note. The quietest is -1 and the loudest is -15. P is the pitch, and must be between 0 and 255. This chooses which note to play. Finally, D is the length of the note, and must be between 1 and 255. Each unit of length is 1/20th of a second, so that D = 20 will give a note one second long. Try the following examples:

```
SOUND 1,-10,100,20
SOUND 2,-10,40,10
SOUND 3,-15,150,40
SOUND 0,-10,5,20
```

For the more complicated types of sound, read your manual about the SOUND and the ENVELOPE commands.

Electron

The Electron uses exactly the same command to produce sounds as the BBC, but it can only play one note at a time. Therefore, you should always put the channel number as 1:

```
SOUND 1,-15,100,20
SOUND 1,-10,40,10
SOUND 1,-15,200,25
```

VIC

The VIC computer does not have simple instructions to make sounds. However, using the POKE instruction you can produce three channels of notes and create noise, and you can set the volume of the sounds. The instructions to use are:

```
POKE 36878,X  to set the volume (0 to 15)
POKE 36874,X  to set note 1 (128 to 255)
POKE 36875,X  to set note 2 (128 to 255)
POKE 36876,X  to set note 3 (128 to 255)
POKE 36877,X  to set 'noise' (128 to 255)
```

Try the following example of making more complicated sounds on the VIC:

```
10 POKE 36877,220
20 FOR L=15 TO 0 STEP -1
30 POKE 36878,L
40 FOR I=1 TO 100 : NEXT I
50 NEXT L
60 POKE 36877,0
70 POKE 36878,0
```

Using colour on your computer

Adding colour to a game program can make it much more interesting to play. You can use colour to make some things show up more than others. You can print messages in different colours — messages of danger in red, for instance, and messages of encouragement in green.

Each computer in this book has different ways of controlling the colour of what is displayed on the screen. The following examples show simple ways in which you can use colour on each machine. If you want to experiment further, then read the relevant parts of your computer manual. You can then go on to using graphics, which you can use to draw pictures on your screen.

ZX-Spectrum

To change the colour of text being printed, use the INK command; for instance:

```
INK 2
```

gives red text. To change the background colour, use PAPER; for instance:

```
PAPER 5
```

gives a pale blue, or cyan, background colour.

BBC and Electron

On the BBC computer, you must choose the mode that you want the screen to operate in. The mode makes a difference to the

number of different colours you can see on the screen at once, and also the number of characters across the screen.

The following examples will work on both the BBC and the Electron. We will use mode 5, which gives you four colours and twenty characters on a line. First, type in the instruction:

```
MODE 5
```

Then try the following examples:

```
COLOUR 1
```

give red text. The other colours, yellow and white can be used by giving the numbers 2 and 3 respectively in the COLOUR instruction.

```
COLOUR 131
```

give a white background to the text. The background number is obtained by adding 128 to its foreground number, so that 129 and 130 give red and yellow backgrounds respectively.

Dragon

The Dragon cannot easily produce text of different colours, although it can draw pictures using its colour graphics. If you wish to try the Dragon's graphics commands, then look them up in your computer manual.

VIC

The easiest way to get colour on the VIC, is to use the keys on the keyboard marked with the various colours. Pressing Control along with a number key will make the computer display text in that colour from then on. Thus:

```
PRINT "#HELLO THERE"
```

where the # sign signifies you pressing Control—3, will make the message appear in red. You can put these control characters anywhere in your text. You can use control—9 and control—0 to get reverse characters.

This glossary gives explanations and examples of most of the common instructions that make up the BASIC language. You can use it to check against the games in this book and to find out more about how they work. In some cases different words are used for the same function on different machines, in which case the alternatives are given.

There is a separate section for BASIC functions, which are not instructions in their own right.

In each section the words are listed in alphabetical order.

BASIC instructions

CLS is a simple command that clears the screen of all the computers except VIC. On VIC you must use:

```
PRINT CHR$(147);
```

DATA is used with the READ command. It tells the computer that the following numbers or strings are just pieces of data that will at some point be READ by the program. You can put more than one item on a DATA statement, by separating each one with commas.

Examples

```
DATA "JAN",30,"FEB",28,"MAR",31
DATA "ONE","TWO","THREE"
DATA 9.8,3.2,77.99
```

DEF FN. . . is used to define your own mathematical instructions or functions. You will probably have noticed that in most of the programs in this book there is a line that looks like:

```
1000 DEF FNR(X)=RND(X)
```

or

```
1000 DEF FNR(X)=INT(RND(1)*X+1)
```

or

```
1000 DEF FNR(X)=INT(RND*X+1)
```

depending on which computer you are using. The right-hand side of this, after the equals sign, is the normal expression for working out

random numbers, but what about the left-hand side? This is called a function definition, and it lets you make up your own mathematical instructions and give them names. Once you have done this, you can use them throughout the program, using the name FNv where v is any normal variable name. The letter X inside the brackets on the left-hand side is not a real variable, but a way of telling the computer what to do with what it finds inside the brackets when you actually use the function. In the above case, typing

```
PRINT FNR(10)
```

will display a random number between 1 and 10. On some computers, this would save typing

```
PRINT INT(RND(1)*10+1)
```

every time you wanted a random number.

Examples

```
DEF FNV(X)=X*1.15
```

(A function to add VAT to the number)

```
DEF FNS(X)=SIN(X/180*3.141)
```

DIM tells the computer to reserve memory for a special type of variable called an array. These variables are really like a lot of boxes in a row; each box can itself hold a number or string. So, you get a lot of places in memory to store numbers, but all the places have the same name. To tell them apart you must put a number in brackets to tell the computer which box you want to look at. You must also decide beforehand the total number of boxes that you want. You put this number in brackets in the DIM statement.

```
DIM A(25)
```

makes an array called A, with twenty-five boxes to put separate numbers in. A(1) is the first, and you could use it like this:

```
LET A(1)=1999
```

A(2) is the next, and could hold something else:

```
LET A(2)=2001
```

You could use all the boxes, or elements, in A to store different numbers, all the way up to A(25)

You can use a loop to print out all the boxes in an array (see the entry under FOR). To do this, you should put the loop control variable inside the brackets after the array name, so that each time the loop is repeated, the next box will be printed. For example:

```
FOR I=1 TO 10
PRINT A(I)
NEXT I
```

DIM can be used to create a grid of boxes, instead of a simple row. This is called a two-dimensional array, and you must give two numbers inside the bracket in the DIM to specify the array's size. You must give two numbers to specify the box whenever you wish to use this array. For example:

```
LET A(2,4)=5
```

would store the number 5 in what may be thought of as the second box in the fourth row of the grid. You can also create an array of strings instead of numbers, by using a string variable name in the DIM instruction.

Examples

```
DIM C(52)
DIM B(8,8)
DIM B$(20,5)
```

END will tell the computer to finish running the program and wait for another command, such as RUN. You can put more than one END command in a program, but too many get very confusing.

The ZX-Spectrum uses STOP, not END. See the entry under STOP.

FOR marks the beginning of a set of instructions that you want the computer to go through a number of times. This repetitive operation is called a loop, because of the way it goes round and round. You must use a variable to control the loop, and a beginning

and end number for this variable. The computer will start with the beginning number, and add one each time it goes round until it reaches the end number. Then it will carry on with the rest of the program. To show the computer where the end of the loop is, you must put a NEXT command. For instance:

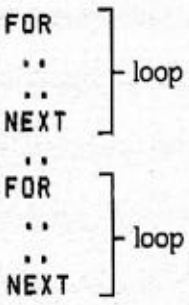
```
FOR I=1 TO 5
 ..
 ..   the part of the program that you want to loop
 ..   round goes here
 ..
NEXT I
```

This loop is controlled by the variable I. It starts at one and increases by one until it reaches five, so the instructions within it will be repeated five times. You can change the amount that the loop control variable increases each time by putting the word STEP into the FOR instruction. Put the number that you want to increase each time after STEP.

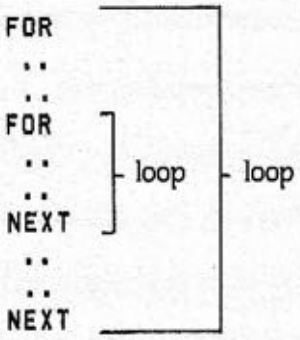
Examples

```
FOR J=10 TO 20
FOR K=1 TO 10 STEP 2
```

You can have loops one after the other, as



or inside each other, as



The latter is called 'nesting'.

Note that the control variables must be carefully specified. Each of your loops will usually use a different control variable name.

GOSUB is a little like GOTO in that it tells the computer to go to a particular line of the program other than the next one. Unlike GOTO, however, the computer remembers the line number that it came from, and will go back to it when it encounters a RETURN instruction. You must always remember to use a RETURN instruction at the end of this section. This section is called a subroutine. Subroutines are useful if you want to do the same thing again and again in different parts of a program. Instead of writing the same instructions again and again through the program, just write them once, say at the end of the program, and use GOSUB to run them whenever you need to.

Examples

```
IF G=9 THEN GOSUB 500
GOSUB 1000
```

GOTO simply tells the computer to carry on running the program from a particular line number. Normally the computer goes through the program line by line. Suppose, however, you want to change what happens during a program, depending on the result of an IF...THEN command. You can use GOTO to make the computer run a different part of the program, but only when a particular thing is true. Alternatively, having executed one particular part of the program, you may want to start again near the beginning of the program. You then use GOTO to transfer control back to the earlier program lines.

Examples

```
IF Z=1 THEN GOTO 1095
GOTO 900
```

IF. . . THEN is used to make decisions. It asks the computer to work out whether some condition is true or false, and tells it to do different things depending on the answer. You can use various things in the expression:

=	equal to
<	less than
>	more than
<=	less than or equal to
=>	more than or equal to
<>	not equal to

AND is used if you want to decide if both one thing AND the other are true
OR is used if you want to decide if one thing OR the other OR both are true.

If the final decision is that the condition is true, the computer will then go on to the instructions following the word THEN. On the other hand, if the condition was false, it will simply go on to the next line of the program.

Examples

```
IF Z=9 THEN GOTO 100
IF N$="JOHN" AND A>19 THEN PRINT "HELLO"
IF (Q/2)<=SIN(Z) OR F=1 THEN LET G=99
```

INPUT is used to get answers from the person using the computer. It is a bit like a LET command, except that it obtains the value to put into the variable from you instead of from the program. You can INPUT values for numeric or string variables, according to the type of the variable you use. For instance:

```
10 INPUT Z
```

will expect you to type in a number, which will be stored in variable Z. You must always press the ENTER or RETURN key so that the computer knows that you have finished typing in the answer and it can carry on with the program.

Examples

```
INPUT Z
INPUT Q$
```

LET is used to put a value into a named place in the computer's memory. These places are like boxes, and are called variables. They are usually given single-letter names, and there are two main types. Numeric variables can hold any number, for instance:

```
LET Q=11
```

will put the number 11 into a variable called Q. String variables can hold any letters, digits or other characters; their names are followed by a dollar sign, for instance:

```
LET N$="HELLO-123"
```

You can also do maths using the LET command, by putting more complicated things on the right-hand side of the equals sign. Remember that on the left must be a single variable name, the name of the variable where you want to put the final result. For instance:

```
LET A=10+(B/2)
```

will take the value stored in variable B, divide it by two, add ten and finally store the answer in variable A. You can also use mathematical functions such as square root and sine, if you need them in your program.

You can use + with string variables. In this case addition means adding one string on to the end of another. For example:

```
LET N$="FRED"
LET M$=" BLOGGS"
LET Q$=N$+M$
```

would make the string variable Q$ contain the string FRED BLOGGS.

Examples

```
LET Q=45
LET G1=10
LET R=(Z+L)/(4*P)
LET X=SIN(T)
LET W$="HELLO"
LET S$="THE "+C$
```

NEXT is the command used with FOR (see entry FOR). It tells the computer where the end of the loop is. Each time the computer comes across a NEXT statement (followed by the name of the loop control variable, e.g. NEXT I) it increases the loop variable by one and jumps back to the line with the FOR command in it, unless it has reached the terminating number.

Examples

```
NEXT W
NEXT
```

The second of these examples is NEXT without the control variable. Provided that the program has the same number of FORs and NEXTs, and that these are placed correctly, the computer will know which FOR to return to. The omission of the control variable is not allowed on the ZX-Spectrum.

PRINT tells the computer to put something on the screen. It can be used to display messages (look at all the programs in this book) and also to show what particular variables have stored in them. To display a message, put it inside quotation marks after the PRINT command:

```
PRINT "HELLO"
```

To show the contents of a variable, put its name after the PRINT command:

```
PRINT N
or
PRINT A$
```

For a blank line on the screen, type in PRINT on its own. You can separate things after PRINT with commas or semicolons. Usually a semicolon will display the items next to each other, while a comma will space them out on the screen.

Examples

```
PRINT Z
PRINT "HELLO THERE ";N$
PRINT F*V,B/2
```

RANDOMIZE is used to get around a problem with random numbers generated by the computer. You may notice that the computer always gives you the same random numbers just after switching it on. To stop this happening, use the following commands, right at the beginning of your program:

```
10 RANDOMIZE          for the BBC and Electron
10 RANDOM             for the ZX-Spectrum
10 LET R=RND(-TI)     for the VIC
10 LET R=RND(-TIMER)  for the Dragon
```

These make sure the random numbers start off completely randomly.

READ is used to copy DATA you may have stored in the program into chosen variables. It behaves like INPUT, but instead of stopping and asking the person using the computer for something, it gets that something from the list of items that you should have put into the program beforehand. These items are put in DATA statements. Each time you execute a READ command, the next item (starting with the first item when you run the program) is read into the variable whose name you give. If you give a numeric variable it will

expect to read a number from the DATA statements, and if you give a string variable it will expect to find a string in quotes. These commands are useful if you want to have a lot of data in a program which is unchanging — for instance, the names of the months and how many days each one has.

Examples

```
READ P
READ N$,A,B
```

RETURN tells the computer that it has reached the end of a subroutine, and that it should return to the program line that the subroutine was called from. It will then continue with the next instruction after GOSUB.

STOP is the same as END, except that on some computers a message is printed to tell you what line the program stopped in. Note that the ZX-Spectrum does not understand END, and so you must always use STOP.

You can also use STOP while you are trying to get a new program to work. By putting it in the middle of the program, you can temporarily stop the computer while you look to see what is contained in the variables, and then use the CONT command to carry on running the program.

BASIC functions

Now we come on to various 'functions' that BASIC understands. These can be used together with variables in LET and IF...THEN commands. Most of them produce a result depending on something in brackets. For the functions that are not described, look up what they do in your computer manual.

ABS ignores any minus sign in front of a number, and always gives you the absolute value. For example:

```
ABS(42)      gives the answer 42, and
ABS(-42)     also gives 42
```

CHR$ converts numbers into single letters or other characters. It uses a special computer code called ASCII, in which each number (normally between 1 and 127) corresponds to a particular letter. For example, the letter A has a code number of 65, so that

```
PRINT CHR$(65)
```

would print out an A.

Examples

```
CHR$(147)
CHR$(64+I)
```

INT is a maths command that is used to convert numbers with something after the decimal point into whole numbers. For positive numbers, it just ignores everything after the point, so that INT(19.86) is 19. For negative numbers, it also ignores everything after the decimal point, and it increases the number by one, making it more negative; thus INT (—5.45) is —6.

Examples

```
INT(S/10)
INT(RND(1)*10)
```

RND is a very useful function. The simplest form produces a random number which is always between 0 and 1. You can use this, and a bit of maths, to get the computer to choose random numbers in any range, or to get the computer to choose randomly between doing two different things.

Examples

```
RND(1)      (VIC, BBC and Electron)
RND         (ZX-Spectrum)
RND(0)      (Dragon)
```

On the BBC, Electron and Dragon computers there is another version of the function RND that will give you a random whole

number between 1 and your selected number. For instance, RND(6) will give a random whole number between 1 and 6.

To get a random whole number between 1 and another number on other computers, you should use the expression:

```
INT(RND(1)*n+1)  (for the VIC)
```
or
```
INT(RND*n+1)     (for the ZX-Spectrum)
```

where n is the highest random number that you want. For example, to get a random number between 1 and 6, use

```
INT(RND(1)*6+1)  (for the VIC)
```
or
```
INT(RND*6+1)     (for the ZX-Spectrum)
```

SIN, COS, TAN, ATN are all maths functions that you can use in your programs. Remember that the computer normally works in Radians, not Degrees.

Examples:

```
SIN(3.142)
COS(A)
TAN(T/2)
ATN(F+L)
```

SQR is a useful function that gives you the square root of a number.

Examples:

```
SQR(16)
SQR(T*2)
```
(would give the answer 4)

VAL takes a string variable and, if there is a number written in that variable, converts it to a true number that you could put into a numeric variable instead. In other words, if your program has a number written into a string variable, and you want to do some maths on it, you must use VAL first.

Examples:

```
VAL("45")    (would give the number 45)
VAL(N$)
```

BASIC direct commands

There are a few commands which are not usually used within a program. These 'direct' commands are used to tell the computer what to do with your program.

LIST is used to display all or part of your program on the screen. Typing LIST on its own will display the whole program (on the ZX-Spectrum, the computer will pause at the end of every page and ask whether you wish to continue). Typing LIST followed by a line number will list the program starting at that line number. To display a smaller section of a program, you should give the start and end line numbers of the section. For example:

```
LIST 10-100
```

will list lines 10 to 100 on the Dragon and the VIC, while

```
LIST 10,100
```

will list lines 10 to 100 on the BBC and Electron.

```
LIST
```

will list from line 10 on the ZX-Spectrum

LOAD is used to recall programs from tape or disc back into the computer. The Dragon uses CLOAD to load programs from tape. Look up this command in your computer manual to find out how to use it.

RUN tells the computer to start running the program in its memory. It will always start at the very first line of the program. Before running the program, the computer clears any variables that have been used previously, so that it's memory does not become full up.

SAVE is used to store your programs onto tape or disc. The Dragon uses CSAVE to save programs onto tape. Look up this command in your computer manual to find out how to give the programs names on the tape or disc.

Here is a list of some of the ASCII* codes. Each character has a Hexadecimal or Decimal number next to it, which is its corresponding code:

Char	Hex	Dec	Char	Hex	Dec
sp	20	32			
!	21	33	>	3E	62
"	22	34	?	3F	63
£	23	35	@	40	64
$	24	36	A	41	65
%	25	37	B	42	66
&	26	38	C	43	67
'	27	39	D	44	68
(	28	40	E	45	69
)	29	41	F	46	70
*	2A	42	G	47	71
+	2B	43	H	48	72
,	2C	44	I	49	73
-	2D	45	J	4A	74
/	2F	47	L	4C	76
0	30	48	M	4D	77
1	31	49	N	4E	78
2	32	50	O	4F	79
3	33	51	P	50	80
4	34	52	Q	51	81
5	35	53	R	52	82
6	36	54	S	53	83
7	37	55	T	54	84
8	38	56	U	55	85
9	39	57	V	56	86
:	3A	58	W	57	87
;	3B	59	X	58	88
<	3C	60	Y	59	89
=	3D	61	Z	5A	90

*ASCII stands for American Standard Code for Information Interchange. Most computers use this code to represent the various different letters, numbers, and symbols used by the computer.

SOFTWARE ORDER FORM

Computer Games

The programs listed in this book are available on cassette direct from the publisher.

There is a cassette available for each machine, price £4.95 (plus £1.00 postage and packing). The cassettes are only available on a cash-with-order basis.

Fill in the form below, making sure you state clearly which machine you have. Send the completed form with your cheque or postal order to:

Sparrow Software
Hutchinson House
17-21 Conway Street
London W1P 6JD

Please send me one cassette at £4.95 (plus £1.00 p+p) for the following machine:

ZX-Spectrum	☐	Electron	☐
VIC	☐	Dragon	☐
BBC	☐	*(tick box where applicable)*	

I enclose cheque/PO made out to Sparrow Books.

Name _____

Address _____

_____

Postcode _____

From laying depth-charges on the ocean bed to battles in the farthest galaxies; from conflicts with creatures unknown since prehistoric times to confrontations with sophisticated robots from a future yet to be imagined – here is an exciting collection of computer games that you can program and play from the very first day that you own your computer.

COMPUTER GAMES

Dan Isaacman

SPARROW BOOKS

ISBN 0-09-933330-9

Children's
Non-fiction
U.K. £1.50

90000

9 780099 333302